Graphical Security Modelling and Assessment for the Internet of Things

---

A thesis

submitted in partial fulfilment

of the requirements for the Degree

of

Doctor of Philosophy

in the University of Canterbury

by

Mengmeng Ge

Supervision Committee

Supervisor: Dr. Dong Seong Kim

Department of Computer Science and Software Engineering

University of Canterbury

2018

Contents

List of Figures

List of Tables

Acknowledgement

I am indebted to many people in making my Ph.D. thesis possible. I would like to express my gratitude to them for their help and support.

First and foremost, I would like to thank my senior supervisor Dong Seong Kim. It has been a great honour to be his Ph.D. student during the past three years. He has not only given me guidance and advice for my research, but also supported me in all aspects of my study and life in New Zealand. I appreciate his time that he has spent on helping and motivating me and his thoughtfulness on respecting my decisions throughout my Ph.D. journey. All his contributions have made the challenging Ph.D. experience productive and delightful. I have also learned from him how to be a professional researcher in my future career.

The members of the Cyber Security Lab have contributed to both my professional and personal time at the University of Canterbury. We have made good collaborations on a couple of papers, constantly encouraged each other and had a lot of fun time both in and outside of the lab. I am especially grateful for the generous help given by the postdoctoral researcher Jin B. Hong. He has provided detailed introduction of the graphical security model and its implementation and given valuable feedback on the scenario selection, problem analysis and algorithm design for the first two topics covered in my thesis. Besides, I would like to thank all the past and current postgraduate and summer students in the lab, Simon, Dilli, Matthew, Sultan, Fangcheng, Paul and Sephy, for many inspired discussions, interesting and informative talks and timely help on any technical problems. I would also like to acknowledge the research interns, Grace and

Ashley, who have come through the lab during the semester and worked with me on my research topic.

I gratefully appreciate the funding sources, the Department of Computer Science and Software Engineering at the University of Canterbury and the G B Battersby-Trimble Scholarship, that made my Ph.D. work possible. I am grateful to all the staff in the department for their technical and mental support in all aspects of my study. The concise formalism of the framework proposed in the thesis would not have been possible without the revision by Walter Guttmann. I appreciate his rigorous attitude towards the details and ambiguities in the paper that he has co-authored with me. Moreover, special thanks to my associate supervisor Moffat for his emotional support (especially the impressive talk with him at the end of 2015 which motivated me to embrace the failures throughout my Ph.D. journey), to my confirmation examiner Andreas for his constructive feedback on my confirmation report and also for accepting me as the tutor of his networking courses and supporting me through the courses, to the senior tutor Yalini for her constant encouragement during the tutor experience and thoughtfulness for arranging gatherings within the department, to the technicians Philip, Steven and Peter, for their generous and timely help on any technical problems arising from my research, to the administrator Alex for solving any general problems whenever I enquired. I am also grateful to the past and current postgraduate students in the department, Geela, Tieta, Caitlin, Enos, Amir, Prerna, Manpreet, Chathrie, Marianne and Saima, for all the interesting talks and fun time both in and outside of the department.

Lastly, I would like to give a big thank to my parents for their enduring love and faithful support in all my pursuits. A special gratitude also goes to my friends in Christchurch, Grace, Nurul and Denise, who have accompanied with me and given me encouragement during the tough times.

Abstract

The Internet of Things (IoT) is enabling innovative applications in various domains and offering convenience in different aspects of people's life. Characterised by the constrained resources, heterogeneous techniques and wide-scale structure, the IoT introduces a variety of known and unknown vulnerabilities that can be exploited by the attackers to break into the systems to conduct malicious activities (e.g., steal sensitive data, compromise the IoT devices). Therefore, protecting the IoT to defend against the potential attacks is of critical importance. The motivation of the thesis lies within the field of the security modelling and assessment for the IoT to mitigate the impact of potential attacks. Current research on the IoT security modelling is very limited due to the pioneering features of the IoT. Besides, there is no previous work on constructing a formal graphical security model (e.g., Attack Graphs (AGs) [127], Attack Trees (ATs) [122]) for the IoT. Additionally, traditional defence mechanisms may not work well in securing the IoT due to the existence of the forever-day vulnerabilities (i.e., non-patchable vulnerabilities) in the IoT devices. Lastly, there lacks an approach that can combine different defence mechanisms in an optimal way to increase the security of the IoT at a reasonable cost. In order to address the above security issues, we have three goals in the thesis, which are: (i) to develop the security assessment framework for the IoT that can model and assess the security of the IoT; (ii) to develop the proactive defence mechanisms to address the security issues arising from the non-patchable vulnerabilities in the IoT devices; and (iii) to develop an approach to optimise the combinations of

different defence mechanisms to improve the security of the IoT under the budget constraint.

To achieve goal (i), we propose a framework for security modelling and assessment of the IoT, named the security assessment framework for the IoT. The driving idea behind the framework is to mitigate the impact of potential attacks in the IoT and increase the IoT security level via the graphical security model along with the evaluation metrics. Generally, the framework consists of five phases: 1) data processing, 2) security model generation, 3) security visualization, 4) security analysis, and 5) model updates. By using the framework, we can identify potential attack paths in the IoT, analyse the security of the IoT using the well-defined security metrics, and assess the effectiveness of different defence mechanisms. Three different IoT deployment scenarios are used to evaluate the framework, which are the smart home, wearable healthcare monitoring and environment monitoring. The analysis results show the capabilities of the proposed framework for capturing potential attack paths in both small-scale and large-scale IoT networks and assessing the effectiveness of the device-centric and network-level defence mechanisms on mitigating the impact of attacks.

To achieve goal (ii), we propose to change the attack surface of the IoT to increase the attack effort with the existence of the non-patchable vulnerabilities in the IoT devices. With the support of software-defined networking (SDN), we develop two proactive defence mechanisms that reconfigure the network topology of the IoT. We implement the reconfiguration algorithms and integrate them with the security assessment framework. We analyse how the security and performance change when the proposed mechanisms are deployed through simulations. The results show our proactive defence mechanisms in the SD-IoT effectively increase the attack effort, while maintaining the performance in terms of the average shortest path length.

To achieve goal (iii), we propose a novel approach to combine the strategic deployment of adaptive deception technology and the patch management solution for the IoT under the budget constraint. We use a graphical security model along

with three evaluation metrics to evaluate the effectiveness and efficiency of the proposed defence mechanisms. We apply the multi-objective genetic algorithm to compute the Pareto optimal deployments of the defence mechanisms to maximise the security and minimise the deployment cost. We present a case study to show the feasibility of the proposed approach and to provide the defenders with various ways to choose the optimal deployments of the defence mechanisms for the IoT. We also compare the runtime and accuracy of the genetic algorithm against the exhaustive search algorithm. The results show that the genetic algorithm is much more efficient to compute a good spread of the deployments compared with the exhaustive search algorithm when the scale of the IoT increases.

In summary, the contributions of the thesis are: (1) the development of the security assessment framework for the IoT to improve the security of the IoT and to mitigate the impact of potential attacks; (2) the evaluation of the framework via various use cases; (3) the development of the defence mechanisms and reconfiguration algorithms that change the attack surface of the IoT under the support of SDN to increase the security of the IoT with the non-patchable vulnerabilities; (4) the development of the approach to compute the optimal deployments of the defence mechanisms for the IoT under the budget constraint.

Chapter 1

Introduction

In the Internet of Things (IoT), every physical object becomes locatable, addressable and reachable in the virtual world [114, 115, 128]. As more and more objects in the physical world are expected to connect to the Internet, the IoT is supposed to contain millions or billions of objects which will communicate with each other and with other entities (e.g., human beings). These objects not only include computers and laptops which already exist in the traditional networks, but also contain physical devices (e.g., home appliances, vehicles, *etc*). They are connected through heterogeneous communication techniques, for example, Wi-Fi [57], ZigBee [72], Bluetooth [96], *etc*. Some IoT devices support multiple communication techniques and are capable of connecting networks using different communication protocols. Additionally, many IoT devices have constrained resources and limited computational capabilities, and are deployed in an open environment (e.g., street lights) [145].

1.1 Problem Statement

The large number of heterogeneous devices offers people convenience, but at the same time brings a variety of known and unknown vulnerabilities. The vulnerabilities reside in various aspects of the IoT systems, including devices

(hardware, operating systems), communication protocols, service applications, service APIs and the design of the IoT architecture, *etc*. By exploiting those vulnerabilities, an attacker can launch various attacks to compromise the IoT, including eavesdropping, Denial of Service (DoS) attacks, physical damage, node capture and controlling [115]. These attacks may have catastrophic effects upon the normal functionality of the IoT. Thus protecting the security of the IoT is a difficult yet important task. With the presence of complex attacks [7, 8], the ability to discover potential attack scenarios (e.g., an attacker's paths to a target IoT device) by utilising the existing vulnerabilities and to mitigate the impact of malicious attacks becomes a critical issue.

As a sequence of the IoT devices with the exploitable vulnerabilities can form the attack paths, it is ideal to remove all the vulnerabilities (e.g., patching software vulnerabilities). However, it is infeasible to do so according to several factors. Firstly, as the IoT contains a large number of devices, it is impossible to remove the vulnerabilities for all devices when taking into account the time, effort and cost. Moreover, forever-day vulnerabilities cannot be removed, and unknown vulnerabilities (e.g., zero-day vulnerabilities [27, 39]) are impossible to patch. In specific, for the forever-day vulnerabilities, vendors and suppliers no longer provide support for their products (e.g., the products have reached their end-of-support phase); the zero-day vulnerabilities refer to the vulnerabilities which are unknown to the vendors that are exploited by the attackers, which leave the vendors with zero day to create patches. Additionally, patching vulnerabilities can be complicated for users with little or no knowledge about security. This requires an alternative approach to mitigate the impact of attacks targeting the IoT while meeting the functionalities and security needs for everyday operations.

When securing the IoT, different combinations of the defence mechanisms (both device-level and network-level) can be applied for the IoT to defend against the sophisticated attackers. Here, the sophisticated attackers refer to the attackers that are capable to collect the intelligence about the target, bypass multiple layers of the defence mechanisms and use multiple exploits to compromise the

target. Thus, given a set of the deployments of different defence mechanisms and a certain budget, selecting the deployments of the defence mechanisms in an optimal way to maximise the security while minimising the deployment cost is a critical issue in the IoT.

1.2 Research Questions and Goals

This thesis takes a security modelling approach to target the following research questions that have not been thoroughly addressed yet:

- Q1: how can we discover the potential attack scenarios by utilising existing vulnerabilities and mitigate the impact of attacks?
- Q2: how can we protect the IoT with the existence of the non-patchable vulnerabilities in the IoT devices?
- Q3: how can we select the optimal deployments of the defence mechanisms to maximise the security of the IoT and minimise the deployment cost?

The goals of the thesis are to advance the security modelling and assessment for the IoT to deal with the security issues arising from the IoT. Three goals corresponding to the research questions are described as follows:

- G1: to develop the framework to model and assess the security of the IoT and to evaluate the framework.
- G2: to develop the defence mechanisms to address the security issues arising from the non-patchable vulnerabilities in the IoT.
- G3: to develop the approach to compute the optimal deployments of the defence mechanisms for the IoT.

1.3 Methodology

We propose the methodology which comprises the systematic analysis of the procedures to solve the research questions. Five phases are introduced when a research question is formed. We describe each phase in detail in the following.

System model: we make the assumptions for the IoT networks which will be analysed based on the real-world networks. The assumptions include the application scenario (e.g., healthcare, smart home), network component (i.e., the types of the nodes in the network), network size (i.e., the number of each type of the nodes in the network), network deployment (i.e., the area of the deployment, network topology) and node specification (i.e., the operating system and applications running on it, communication protocol, vulnerability information). The abstraction of the real-world systems aims to provide the information which will be included in the models as accurate as possible. The assumptions for the IoT networks in a specific scenario are used to validate the models or the frameworks but do not limit the scope of the research question.

Attacker model: we make the assumptions for the goals and capabilities of the attackers and the potential attacks that can be carried out by the attackers. The assumptions can be obtained from the vulnerability information of the nodes in the IoT networks and the real-world attacks in which these vulnerabilities are exploited. In specific, the attackers' capabilities include the access to the network (i.e., a remote attacker from the Internet or an attacker located in the local network), the attack tools, the exploitation of the vulnerability information and the gained privilege after the exploitation. As the attacker model is required by the graphical security model, we also need to specify the potential entry points and targets for the attackers.

Defence model: we develop the defence model which comprises the mechanisms or approaches to improve the security of the IoT networks. The mechanisms can be in both device-level and network-level. They are designed and implemented to solve the research question and validated based on the pre-

defined system model and attacker model.

Framework: we develop the framework which specifies the workflows to compute the expected results via the graphical security model and the evaluator. The framework incorporates the system model and attacker model to provide the inputs into the graphical security model and the evaluator. Some network features that are defined in the system model need to be provided as the inputs into the graphical security model and the evaluator, which include the network topology information, vulnerability information for each node and security metric values for each vulnerability. The evaluator has a list of the metrics, including different security metrics and performance metrics to assess the networks. The framework can also incorporate the defence model in order to define the procedures for the assessment of the effectiveness of the defence model. New metrics will be designed in the evaluator to assess the defence model.

Evaluation: we carry out the evaluation via the use cases, case studies or simulations. The evaluation is to validate either the framework or the defence model. Besides, the scenarios of the use cases, case studies and simulations are based on the system model and attacker model.

We describe the usage of the methodology to deal with each proposed research question in the following.

- For the research question Q1, we develop the framework at first and then evaluate the framework via different use cases. For each use case, we define the system model and attacker model as the inputs of the graphical security model and the evaluator to perform the security assessment and also provide the defence model to demonstrate the usage of the assessment of the defence mechanisms.

- For the research questions Q2 and Q3, we define the system model and attacker model at first. Afterwards, we develop the defence model and define the procedures in the framework to validate the defence model via the simulations.

1.4 Research Contributions

Four primary contributions are proposed in the following to the graphical security modelling and assessment for the IoT.

1. *Development and formal definition of the security assessment framework for the IoT:* we propose a framework for security modelling and assessment of the IoT (the work has been published in [48, 50]). The framework is used to construct a graphical security model (in particular, a scalable security model named Hierarchical Attack Representation Model (HARM) [58]) and a security evaluator to automate the security analysis of the IoT. The security evaluator uses various security metrics to assess the security and interacts with an analytic modelling and evaluation tool, Symbolic Hierarchical Automated Reliability and Performance Evaluator (SHARPE) [121], to output the analysis results.

2. *Evaluation of the security assessment framework via different use cases:* we evaluate the framework using three use cases, including a smart home, wearable healthcare monitoring and environment monitoring (the work has been published in [48, 50]). In each use case, we create the example IoT network, construct the graphical security model, carry out security analysis based on the calculated metrics and assess the effectiveness of the defence mechanisms.

3. *Security analysis of the software-defined IoT with the non-patchable vulnerabilities:* in the cases that we cannot remove the vulnerabilities, we can change the attack surface of the IoT to increase the attack efforts (the work has been published in [49]). Recently, software-defined networking (SDN) is foreseen as a key enabler for the IoT as the SDN is able to manage large-scale networks, establish complex routing topologies and simplify user operations [63, 76, 105, 126]. In particular, it centralises the network control and provides a dynamic, flexible and automated reconfiguration

of the networks. Current work has already shown the possibility and feasibility to integrate the SDN with the IoT (Section **??**). We reconfigure the network topology of the IoT based on the SDN functions and conduct security and performance analysis via the security assessment framework.

4. *Computation of the optimal deployments of the defence mechanisms:* we consider two defence mechanisms to increase the security of the IoT networks and compute the optimal deployments of these defence mechanisms using the security assessment framework with the addition of the multi-objective optimisation algorithm (the work has been submitted to a conference A). The defence mechanisms include (1) the deception technology to divert the attackers from the real assets and (2) the security patch management solution to reduce the attack surface.

In summary, the security assessment framework with the additional reconfiguration and optimisation modules is proposed to model and assess the security of the IoT, to increase the security of the IoT with the non-patchable vulnerabilities under the support of SDN and to compute the optimal deployments of the defence mechanisms for the IoT. Use cases and case studies are introduced to demonstrate the viability of the framework and simulations are performed to evaluate the proposed mechanisms and algorithms.

1.5 Thesis Structure

The rest of the thesis is organised as follows. Chapter 2 discusses the current research in the field of security modelling and assessment for the IoT. Chapter 3 presents the description and formal definition of the framework for modelling and assessing the security of the IoT and the evaluation of the framework via three use cases (addressing the research goal G1). Chapter 4 proposes the security analysis of the software-defined IoT with the non-patchable vulnerabilities (addressing the research goal G2). Chapter 5 provides the computation of the optimal defence

mechanisms for the IoT under the budget constraint (addressing the research goal G3). Chapter 6 discusses the usability and limitations of the thesis and points out the future research directions. Chapter 7 concludes the thesis.

Chapter 2

Literature Review

In this chapter, we first begin with discussing current work on security models for the non-IoT networks and IoT in Sections 2.1 and 2.2 respectively. In Section 2.3, we discuss the current SDN solutions for the IoT to investigate the viability of integrating SDN with the IoT. In Section 2.4, we present the work addressing the security issues of the SD-IoT. Then we present the current approaches on the optimal selection of the defence mechanisms for the IoT in Section 2.5 and investigate the cyber deception technologies in the IoT in Section 2.6. In Section 2.7, we summarise the main problems that require further investigation and point out our approaches.

2.1 Security Models for the Non-IoT Networks

Graphical security models have been widely used for the security analysis in various types of the non-IoT networks. We discuss the tree-based models, graph-based models, HARMs and related model generation tools.

Tree-based models: Mauw *et al.* [93] proposed a formal representation of ATs including the definition, transformations between ATs, calculations of attribute values associated with ATs and the projection algorithm applied to answer questions of people's interests (e.g., which attack causes damage over a certain

limit).

Ten *et al.* [133] presented an analytical approach to evaluate the vulnerabilities in the supervisory control and data acquisition (SCADA) system using ATs. An AT was constructed according to attack goals and used to evaluate vulnerability indices for each attack leaf, each intrusion scenario and the overall system based on security conditions, countermeasures and password policies. A case study for the power system control network was conducted to identify possible break-in points and to evaluate the vulnerabilities.

Saini *et al.* [122] proposed the idea of threat modelling using ATs. They constructed an AT for an online certificate repository in the Grid Security Infrastructure toolkit and analysed possible attacks and impacts caused by the attacks.

Roy *et al.* [117] proposed attack countermeasure trees (ACTs) for the security analysis by taking into account both attacks and defence mechanisms. In the ACT, defence mechanisms can be deployed on any node of the tree instead of only leaf nodes. Qualitative and probabilistic analysis can be performed using the ACT to evaluate the security of the network. Besides, structural and Birnbaum importance measures can be used to prioritise attacks and countermeasures respectively. They implemented the ACT in the SHARPE and showed the usability of their model in three case studies: ACTs for a BGP attack, a SCADA attack and a malicious insider attack.

Graph-based models: Jha *et al.* [68] proposed an algorithm to generate AGs using a model checking technique for vulnerability analysis. Their algorithm can compute all potential attacks and contain only relevant states of the network and the intruder. They also designed minimisation analysis approaches on attack graphs to formalise the security analysis and incorporated probabilities into AGs to perform reliability analysis.

Ou *et al.* [107] developed a vulnerability analysis tool to analyse the security impact of software vulnerabilities on networks. The tool automatically processes

bug reports from existing vulnerability scanners and generates AGs to perform the security analysis. The tool was implemented on Red Hat Linux. They tested the tool in a testbed with 500 Linux hosts connected via the Internet. The results showed their tool ran efficiently and identified a policy violation caused by vulnerabilities.

Ingols *et al.* [65] improved the AG generation tool designed in [66]. They considered client-side vulnerabilities, zero-day vulnerabilities and two common countermeasures including personal firewalls and intrusion prevention systems. They also redesigned the methods for computing network reachability to support reverse reachability (i.e., compute reachability from the malicious server backwards to the vulnerable clients). The experiments were carried out using a real network with 85 hosts and larger simulated networks. The results demonstrated that the enhanced tool is as scalable as their tool in [66]. Their future work includes modelling additional countermeasures, attacks and adversaries, and performing field tests.

Albanese *et al.* [16] used AGs to efficiently generate network hardening solutions. They defined a network hardening strategy as a set of atomic defence actions and introduced a cost model which takes into account the cost of interdependent actions. Then they designed an approximation algorithm to compute the minimum-cost hardening solution. The experiments were carried out using synthetic attack graphs and the results validated the performance of their approach. The evaluation of the proposed approach using real attack graphs will be included in their future work.

HARMs: Hong *et al.* [58] developed the two-layer graphical security model called HARM to assess the security of enterprise networks. The HARM is generated using network topology information in the upper layer and host vulnerability information in the lower layer. They performed complexity analysis on the HARM, AG and AT and concluded the HARM has smaller or equal computational complexity in the model construction, evaluation (i.e., calculation of attack paths) and update (e.g., host addition or removal) phases of the security

analysis.

Hong *et al.* extended the previous paper and performed the scalability analysis of the multi-layered HARM in [60]. They compared the two-layered and three-layered HARMs with the single-layered AGs in terms of the model construction and evaluation. The simulation results demonstrated that the HARM is more scalable than the single-layered AG. In particular, the three-layered HARM was found to be more scalable than the two-layered HARM.

Jia *et al.* [69] developed a software tool to generate AGs and HARMs from scanning reports, and to convert existing AGs into HARMs. They also designed a visualisation tool to visualise AGs and HARMs. The feasibility of the tool was evaluated using an example enterprise network. Their future work includes supporting different types of HARMs (e.g., using ATs in the upper layer and AGs in the lower layer) and improving collection of network reachability information.

2.2 Security Models for the IoT

Several papers focus on developing the security modelling approaches for the IoT. We discuss them in three aspects: security frameworks, game-based security modelling and adaptive security models.

Security frameworks: several papers proposed a framework for the security modelling and analysis of the IoT. Most papers only presented a high-level description or a theoretical framework without any evaluation work [94, 113, 130, 144] or with incomplete analysis [18, 19, 64]. Some papers performed analysis or simulations to validate the framework [15, 101, 102]. However, the proposed frameworks in the papers require the domain-specific knowledge of the IoT which complicates the model construction. Besides, there is also a scalability issue in [15].

Radomirovic [113] proposed a dense IoT model along with a Dolev-Yao adversary model to address security and privacy issues of communication

protocols in the IoT. The dense IoT is defined as an asynchronous communication network with high connectivity and ubiquitous functionality. An attacker model is also introduced in which the adversary has corruption and fingerprinting abilities. The paper pointed out future work towards a formal model limiting the adversary's capabilities.

Yang *et al.* [144] presented a high-level security framework for the IoT. The framework is based on a model encompassing three interlinked elements, which are communication, control and computation. They regarded the IoT as the linkage between control and computation. The computation algorithms have a direct influence on the end devices. As the direct control can be intervened by attackers, they put security control between computation and control. They concluded protecting IoT is not only a technical issue but also a social issue.

Stepanova *et al.* [130] proposed a theoretical framework for modelling the IoT security based on graph theory. By defining the IoT as "net of nets of things", they designed formalised network property indicators to assess the sustainability of nets of things (NoT) and described a method to maintain the sustainability of the NoT entities. Their future work includes the efficiency evaluation of the method with pre-defined indicators.

Atamli *et al.* [19] provided a threat model which consists of three sources of threats and eight types of attack vectors to determine where efforts should be invested to secure systems. Using the threat model, they analysed the impact of threats and deduced the security and privacy properties for the IoT based on three use cases: power management, smart car and smart healthcare system. Their future work includes the design of a security package that can be used for any use case.

Huang *et al.* [64] proposed a security framework named SecIoT under the 5th generation wireless system. SecIoT consists of a secure authentication system, which employs the multi-channel security protocol for device authentication, a role-based access control mechanism with fine-grained roles, and a risk indicator

interface based on security risk analysis techniques. A prototype IoT was presented with an authentication protocol analysis and user acceptance studies on access control and risk indicator. The user studies indicated that a fine-grained role-based access control should be supported in the IoT and that a risk tree map is the best way to represent risks. Their future work includes the development of availability enhancement and trust management into the framework.

Ashraf *et al.* [18] proposed an overview of threat mitigation approaches in the IoT based on "autonomic security". They classified these approaches into self-protecting, self-healing and hybrid of self-protecting and self-healing, and discussed their usage against different threats in three layers: machine-to-machine, network and cloud.

Mohsin *et al.* [101] designed and implemented a formal framework for the security analysis of the domain-specific IoT networks. The framework models the behaviours of the IoT devices, user-defined policies (e.g., IF-THIS-THEN-THAT) and threats using the Satisfiability Modulo Theories logics to discover the hidden attack vectors. They evaluated the framework via a Building Management System. Their future work mainly includes the visualisation of the threat verification and risk analysis of the IoT against various threats.

Mohsin *et al.* [102] proposed a risk verification framework which can be used in conjunction with the framework in [101] to automate the security analysis for the IoT. The framework takes the inputs of the IoT configurations, operational policies, vulnerability information and attacker capabilities to generate the Markov Decision Process model and then computes the risk exposure probabilities for the configurations using the Markov model and probabilistic model checking. They evaluated the framework via a home security scenario to show that the framework can prioritise the configurations based on risk exposures.

Mavropoulos *et al.* [94] developed a software tool to visualise the IoT systems which facilitates the security analysis. The tool is based on the APPARATUS

framework [95] which identifies the IoT assets and threats during the design phase and vulnerabilities during the implementation phase.

Agadakos *et al.* [15] proposed a formal way to model the cyber and physical interactions of the IoT devices under user-defined policies to reveal potential attack paths. They implemented the proposed approach using Alloy which is a specification language and a structural modelling tool. Three IoT use cases were introduced to evaluate the viability and efficacy of the approach. The performance analysis was also carried out and the results show the approach only works well with small-scale IoT networks.

Game-based security modelling: several papers addressed game-based security modelling for the IoT. However, their scope either focused on mitigating the impact of certain attacks [35] or emphasised model solutions for specific domains [56, 82, 116].

Hamdi *et al.* [56] constructed a Markov game-theoretic model to support decision making in the realm of IoT healthcare applications. Specifically, for smart things, the decision of whether or not to authenticate a forwarding packet is based on the assessment of power life, channel bandwidth, memory capacity and compromised nodes through the game-based model. The performance of the model was evaluated through simulation which showed smart things extend their lifetime by adopting the adaptive security policy.

Chen *et al.* [35] proposed a fusion-based defence mechanism to mitigate the impacts of intentional attacks in the IoT architecture. They formulated a zero-sum game between the defence strategy and the attacker. In the worst-case scenario, the attacker knows the network topology and is capable of compromising all nodes simultaneously. The results of performance evaluation showed the robustness of the IoT was greatly enhanced by the proposed mechanism.

Rontidis *et al.* [116] developed a decision support method which minimises security risks in the field of IoT prosumer selection. A prosumer offers applications or services in the IoT service deployment stages. They formulated

a non-cooperative and complete information game between the user and the attacker. The worst-case scenario was considered where the attacker knows all security controls of prosumers. Following this scenario, a mixed strategy was proposed to randomise the prosumer selection in an optimal way and compared with two heuristic solutions through simulation which proved the effectiveness of the strategy in mitigating security risks.

Lee *et al.* [82] proposed a game theory-based approach for the security analysis of social IoT networks. They identified the attack and security actions via the attack tree, modelled the interactions between the attacker and security administrator using a non-cooperative game and then calculated the payoffs based on the cost of actions and attack impact. The case study in a smart home network was performed to validate the effectiveness of the approach.

Adaptive security models: since 2012, adaptive security has been utilised in the Adaptive Security for Smart Internet of Things (ASSET) project which aims at developing the risk-based adaptive security methods and mechanisms for the IoT. Adaptive security refers to a security solution that learns and adapts to changing environments dynamically, and identifies and responds to the unknown threats. As the IoT is a dynamic system, security mechanisms implemented in the IoT should adapt to the dynamic context. There are a number of papers published from the project. However, their solutions were only designed for the eHealth domain.

Savola *et al.* [125] investigated security objectives of the IoT applications in an eHealth scenario and proposed the definition of a high-level adaptive security management mechanism using security metrics. The proposed mechanism is a cyclic process consisting of four critical models, which are adaptive security monitoring, analytics and predictive models, decision-making models, and metrics-based adaptive security models.

Abie *et al.* [14] introduced the adaptive framework with the emphasis on the adaptive risk management. Based on the continuous cyclic process, the

framework provides security solutions adaptively upon the estimations of risk damage and benefits and evaluates solutions through the security metrics. A patient-monitoring case study was indicated to be used for validating the framework in the future simulation experiment.

In order to accurately evaluate the adaptive security solutions (e.g., [14]) in real-life scenarios and realistic simulations, Berhanu *et al.* [22] presented a design and implementation of a testbed with heterogeneous biomedical sensors (Shimmer nodes and Raspberry Pi Mini-PC with eHealth sensor shields). The testbed was set up using off-the-shelf hardware and open source software and then validated using the impact of antenna orientation on the energy consumption of sensors. The experimental results showed that the testbed functions well, thus being useful in studying the feasibility of the adaptive solutions.

Torjusen *et al.* [136] investigated the run-time adaptive behaviour which deviates from the normal activities of the system. It was regarded as a major threat to the sustainability of the IoT-enabled eHealth services. Based on the risk-based adaptive security framework in [14], they developed a self-adaptive security framework by introducing run-time verification methods. Four run-time verification enablers were integrated into every phase of the initial feedback loop, which are models at run-time, requirements at run-time, dynamic context monitoring and a runtime verification component. The new framework was instantiated by means of Coloured Petri Nets.

Hamdi *et al.*'s work mentioned above (i.e., [56]) is also part of the ASSET project. In their future work, they considered the implementation of the game-based model in the testbed [22].

Habib *et al.* [54] identified assets, vulnerabilities and threats for eHealth applications and proposed a threat detection and prevention mechanism based on adaptive security. Generally, security events are at first gathered by sensors and monitoring components in devices, then analysed to determine whether the events are threats or not. From the analysis, a planning function decides actions

on the events via a knowledge base or learning mechanism. Thus, based on their analysis, the adaptive security mechanism is able to adjust security levels according to the threat levels.

2.3 SDN Solutions for the IoT

Current SDN solutions for the IoT were proposed in three aspects, which include the management of sensing devices in the WSNs, management of user devices in the mobile networks [43] and management of smart devices in the IoT architectures.

SDN for the WSNs: solutions were designed for managing sensing devices inside a WSN. Most solutions were based on flow tables [46, 47, 89, 90, 134, 137] while one solution provided reconfigurable sensors [100].

Mahmud *et al.* [90] introduced the OpenFlow-based sensor, named "flow-sensor", to address the reliability issue of the sensor networks. The flow-sensor uses a control interface to exchange control packets with the Controller and maintains the flow tables for communicating with access points and other flow-sensors. They carried out simulations to validate their design. The simulation results showed that flow-sensors generate less number of packets and use less time compared with traditional sensors.

Luo *et al.* [89] proposed an architecture of a software-defined WSN to address the issues in WSN. They designed a Sensor OpenFlow protocol for the communication between the control plane and data plane based on the OpenFlow specification (v1.3.0).

Gante *et al.* [47] developed a framework for a software-defined WSN. In the framework, sensor nodes forward or drop packets based on a set of pre-installed rules, known as the flow table. The flow table is generated by a logically centralised Controller which was assumed to be implemented as part of the base station. A generic architecture of the base station was proposed to describe the

reconfiguration abilities of the base station in the SDN paradigm. Their future work includes the investigation of multiple Controllers and the communication protocols and the implementation of the proposed framework.

Trevizan de Oliveira *et al.* [137] designed and implemented a TinyOS-based and hardware independent architecture for software-defined WSNs. The architecture consists of SDN-enabled sensor nodes and one or multiple SDN Controllers. They implemented the SDN-enabled sensor node in the TelosB mote and the Controller as a software. They carried out experiments on COOJA simulator and measured delay time of packet forwarding. Their future work includes the measurement of energy consumption for packet forwarding and implementation of a manager application server providing APIs for developing applications.

Miyazaki *et al.* [100] proposed a prototype system of a software-defined WSN. The system consists of a role generation and delivery mechanism and reconfigurable sensor nodes. In the mechanism, roles are generated based on the user-defined scenario description and delivered to sensor nodes. They implemented the mechanism in a server PC connected with 16 reconfigurable sensor nodes and a base station. The experiment results demonstrated that the role assignment can be applied to all sensor nodes within a reasonable amount of time.

Galluccio *et al.* [46] proposed and implemented a stateful SDN solution for WSNs. The solution decreases the exchanged data between the sensor nodes and the Controller and provides APIs for designing new applications. They used a simulated network on OMNeT++ simulator. They also carried out experiment in a testbed with 5 sensor nodes and 1 sink and analysed the network performance through the round trip time, payload and Controller's response time. Their future work will address security issues of their solution.

Theodorou *et al.* [134] used SDN to improve the topology control for the WSNs. They designed two topology control algorithms and implemented them

as modules in the Controller in a testing framework. The testing framework also consists of COOJA simulator and a platform connecting the Controller and the data plane. Their experimental results show that the algorithms can reduce the latency in the topology discovery phase and achieve good performance in the flow establishment. Their future work mainly includes the analysis of heterogeneous and mobile IoT scenarios and experiments in the real testbeds.

SDN for the mobile networks: several SDN solutions were provided in heterogeneous wireless networks to support the end-to-end flows. The wireless networks include future carrier networks, wireless access networks and radio access networks.

Pentikousis *et al.* [109] developed a software-defined mobile network architecture for future carrier networks. The architecture consists of the MobileFlow forwarding engine (MFFE) and MobileFlow Controller (MFC). MFFEs forward packets to the IP/Ethernet-based transport network using flow rules received from the MFC. They validated the flexibility and programmability of the architecture in the On-Demand Mobile Network prototype testbed.

Lei *et al.* [83] proposed a framework of a software-defined wireless access network for an open campus WLAN. The framework introduces a logically centralised Controller, multiple physical access points (APs) and software access points (SAPs) running on top of APs. Each AP can have multiple SAPs to manage AP associations of user equipments. They presented use cases of the framework in two network applications, seamless handover and load balancing. Their future work includes optimisation of the handover and load balancing algorithms.

Bernardos *et al.* [24] developed a SDN-based mobile network architecture for a mobile network operator. In the architecture, multiple heterogeneous radio access networks (RANs), including the UTRAN (UMTS), EUTRAN (LET) and a Wi-Fi hotspot, are connected to a transport core network. RANs and the transport core network are composed of programmable devices. The logically centralised SDN Controller communicates with northbound interfaces (service

providers, operators) and southbound interfaces (RANS, the core network). They presented a case study to describe the interactions between the components in the architecture when the user equipment is attached to a RAN, moves to another RAN and requests a service.

Wu *et al.* [142] proposed a software-defined system architecture for mobile management and flow control in the IoT multi-networks. In the architecture, the IoT devices connect to heterogeneous access points in different partitions (geographical areas) of the network and request data from data servers. Distributed SDN Controllers are connected via partially interconnected switches and support two types of flows: flows between the data servers and the IoT devices, flows between the IoT devices in different partitions. Based on this architecture, they proposed methods for mobility management and flow scheduling and performed simulations and experiments in the testbed. The results show that the SD-IoT system achieves satisfactory and stable performance in flow control and mobility management.

SDN for the IoT architectures: a few papers focused on managing smart devices in the software-defined IoT architectures. However, there lacks simulations or experiments to validate the architecture.

Hakiri *et al.* [55] proposed an IoT architecture that integrates SDN and message-oriented publish/subscribe data distribution service (DDS) middleware. In this architecture, smart devices connect to SDN-enabled IoT gateways which then connect to SDN forwarding devices. A SDN Controller communicates with SDN forwarding devices via southbound APIs. IoT applications communicate with the Controller via the DDS middleware. They discussed several open research issues addressed by the proposed architecture (e.g., mobility, scalability).

Liu *et al.* [85] proposed a SD-IoT architecture for smart urban sensing. This architecture consists of three layers: the physical layer is composed of sensor platforms, gateways, forwarding devices and servers in the Cloud and

is responsible for gathering and transmitting data in an urban environment; the control layer is composed of multiple Controllers (sensor controller, network controller and Cloud controller) for managing physical devices via the southbound APIs and providing data to applications in the application layer via the northbound APIs. They used a case study and a quantitative analysis to show the benefits of the SD-IoT.

2.4 Security Issues of the SD-IoT

There are several papers dealing with the security issues of the SD-IoT. Most papers proposed the approaches to defend against the network-level attacks [31, 33, 124, 143] while one paper focused on the device-level protection [104].

Chakrabarty *et al.* [33] proposed a secure communication protocol by using SDN for the IoT. They used the concept of Black Network to encrypt both the meta-data and the payload in the link layer and network layer. The SDN Controller acts as a trusted third party with a global network view and communicates with the resource-constrained IoT devices via Black packets. They conducted simulations using different topologies and node states (i.e., asleep or awake mode). The results show that the architecture provides a higher level of security while complicates routing compared with existing 802.15.4 protocols. Their future work mainly includes the improvement of routing mechanisms and methods for securing the link layer frame.

Sandor *et al.* [124] introduced a hybrid network architecture for the IoT and designed an algorithm to switch between redundant paths using SDN against DoS attacks. The hybrid architecture consists of SDN switches and non-SDN segments with redundant communication entry points. The algorithm performs automatic switching between redundant entry points by using SDN switches. They conducted experiments to evaluate the performance of the reconfiguration capabilities of the hybrid architecture and the impact on the resilience of the IoT

communications. The results show that the architecture mitigates the impact of a DoS attack on a non-SDN part of the network.

Bull *et al.* [31] developed a SDN-based gateway for the traffic analysis of the IoT devices. The SDN-based gateway consists of the basic SDN switch function for data forwarding, statistic manager for data collection and mitigation actions upon detection of attacks. They implemented the gateway using the POX controller and carried out TCP and ICMP flood attacks to demonstrate the usage of the gateway. Their future work includes testbed development and abstraction of the component as a service.

Nobakht *et al.* [104] proposed a host-based intrusion detection and mitigation framework to capture attacks in the smart home networks. The framework uses machine learning techniques to analyse the traffic of the IoT devices and OpenFlow rules to block or redirect malicious traffic launched from the compromised devices. They implemented the framework as a module of the Floodlight controller and performed experiment through a smart lighting system.

Xu *et al.* [143] proposed a mechanism to defend against the new-flow attack in the SDN-based IoT (in which the IoT devices send packets to the SDN switches instead of the IoT gateways). The new flow attack refers to the attack that exhausts the SDN switches. The mechanism includes a low-cost monitor method to detect the suspicious flow burst and a mitigation method to redirect the malicious flows. They performed simulations and implemented the mechanism as an application on the OpenDaylight controller in the testbed to validate the feasibility of the approach.

2.5 Security Optimisation for the IoT

There is only a few work on the optimal selection of the defence mechanisms for the IoT.

Rullo *et al.* [118] proposed an approach to reasonably allocate security

resources for securing the IoT networks. They modelled the interactions between the defender and attacker as a Stackelberg game and computed the best security resource allocation plan by minimising the maximum risk, maximum criticality, energy consumption and allocation cost. Three scenarios were introduced and the experiment was carried out for one scenario of the wireless sensor networks with different security tools, including installation of an IDS on network node, addition of highly sensitive transceiver and use of tamper resistant sensor nodes.

Rullo *et al.* [119] proposed another approach to compute the optimal security resource allocation plan for the IoT networks with mobile nodes. A risk metric in economics was introduced and a genetic algorithm was used to evaluate the allocation plans with the measure. They simulated a selective forwarding attack and measured packet delivery rate via the proposed approach.

Both of the work only consider device-level mechanisms but lack the evaluation of network-level defence mechanisms. Besides, they made the assumption that the attacker needs to compromise the security resource to carry out further attacks. However, in real world cases, sophisticated attackers are able to remain undetected and compromise the IoT devices directly.

2.6 Cyber Deception in the IoT

Several papers were proposed in the development of the honeypot technology for the IoT.

Pa *et al.* [108] developed an IoT honeypot to emulate the IoT devices and to capture Telnet-based attacks and designed the IoT sandbox to analyse these attacks against the IoT devices running different CPU architectures. They identified four families of malware that target Telnet-enabled IoT devices. Their future work includes the extension of the IoT honeypot and the sandbox to support more protocols and architectures respectively.

La *et al.* [77] introduced a game theoretic model to analyse the security of

honeypot-enabled IoT networks. They assumed that the attacker may deceive the defender with suspicious or seemingly normal traffic and used honeypot-enabled intrusion detection component which reroutes the suspicious traffic to the honeypots as the defence mechanism. The interaction between the attacker and defender was modelled as a Bayesian game with incomplete information. They presented numerical results to verify the proposed model.

Anirudh *et al.* [17] used honeypots for the online servers for mitigating DDoS attacks launched from the IoT devices. They carried out simulations via a client and server model implemented in Python. Their future work includes the validation of the approach in the real-time environment and the analysis of more types of attacks.

Dowling *et al.* [42] created a honeypot that simulates a ZigBee gateway and used it to capture attacks for further analysis. They identified that most captured attacks are Distributed Denial of Service (DDoS) attacks and bot malware.

2.7 Summary

From the current security modelling approaches for the IoT in Section 2.2, there is no previous work on discovering the potential attack scenarios in the IoT in order to protect the IoT under the various attacks. In this thesis, we focus on constructing a formal graphical security model along with the security evaluator to capture the potential attack paths and analyse the security of the IoT without or with different defence mechanisms. There are several benefits of using a graphical security model. Firstly, all potential attack paths can be captured in the model, whence solutions are no longer limited to defending against specific attacks. Moreover, the formal model can be used to analyse the security of various IoT scenarios. Lastly, it provides an intuitive way to analyse the security weaknesses of systems and to evaluate the potential countermeasures because the sequences of the attackers' actions are captured in the model. The outcomes

include a security assessment framework that can capture potential attack paths in the IoT via the graphical security model and analyse the security of the IoT via the security metrics and a comprehensive evaluation of different IoT networks in various scenarios.

From the current SDN approaches for the IoT security in Section 2.3, there is no previous work on improving the security of the IoT with non-patchable vulnerabilities under the support of SDN. After investigating the current SDN solutions for the IoT in Section 2.4, it is feasible to manage different types of the devices in three scenarios: sensing devices in the WSNs, user devices in the mobile networks and smart devices in the IoT architectures. However, there is a lack of unified SDN solutions for controlling the heterogeneous end devices in the IoT architecture. As the WSNs play a key role in the connectivity of the smart objects in the IoT and there are already several papers on designing and implementing the software-defined sensor networks, we design proactive defence mechanisms for the software-defined sensor networks in the IoT scenarios, develop the reconfiguration algorithms to change the network topology and analyse the security and performance of these networks via the graphical security model and the security evaluator. The outcomes consist of the development of the defence mechanisms and evaluation of the mechanisms via the security assessment framework.

From the current optimisation approaches for the IoT in Section 2.5, there is no previous work on combining different defence mechanisms and evaluating the effectiveness of these mechanisms on the IoT via the graphical security model. After investigating the cyber deception technologies for the IoT in 2.6, there is no research on the analysis and evaluation of the modern deception technology for the IoT. In this thesis, we propose to use the graphical security model along with the evaluation metrics to evaluate the effectiveness of different deployments of the defence mechanisms (including the modern deception technology and patch solution) and apply the multi-objective genetic algorithm to compute the optimal deployments of the defence mechanisms. The outcomes consist of

the development of the evaluation metrics for the defence mechanisms and the optimisation problem for the IoT security, and the evaluation of the optimisation approach.

Chapter 3

Security Assessment Framework for the IoT

Graph-based and tree-based security models (e.g., attack graphs (AGs) [127], attack trees (ATs) [122]) have been widely used to assess the security of various systems [61, 75]. In the graph-based attack models, an AG shows all possible sequences of the attackers' actions that eventually reach the target. With the increasing size of the network, calculation of a complete AG has exponential complexity, thus causing a scalability issue. In the tree-based attack models, an AT is a tree with nodes representing the attacks and the root representing the goal of attacks. It systematically presents potential attacks in the network. However, an AT also has the scalability issue when the size of the network increases.

In order to address the above issues, the two-layered HARM [58] was introduced, which can combine various graphical security models onto different layers. In the two-layered HARM, the upper layer captures the network reachability information and the lower layer represents the vulnerability information of each node in the network. The layers of the HARM can be constructed independently of each other. This decreases the computational complexity of calculating and evaluating the HARM compared with the calculation and evaluation of the existing single-layered graphical security

models. Thus, the two-layered HARM addresses the scalability problem of the single-layered AG and AT. To further improve the scalability, the three-layered HARM was developed in [60] with the subnet reachability at the highest layer. We use the three-layered HARM as our graphical security model in the security assessment framework for the IoT.

In this chapter, we discuss current work on the security modelling for the IoT, present the description of the security assessment framework for the IoT, the formal definitions of the framework and the graphical security model (i.e., the three-layered HARM), the detailed calculation steps of security metrics and a comprehensive evaluation using both heterogeneous and homogeneous IoT networks to validate the framework.

To the best of our knowledge, this framework is the first approach to use a graphical security model in modelling and assessing security for the IoT. The main contributions are summarised as follows:

- Propose a framework for modelling and assessing security of the IoT (Section 3.2);
- Develop a scalable graphical security model to compute attack scenarios (Section 3.2);
- Formally define the framework (Section 3.2.2);
- Use various security metrics to carry out the analysis (Section 3.2.2.3); and
- Evaluate the framework using three scenarios, including a smart home, wearable healthcare monitoring and environment monitoring (Section 3.3).

3.1 Introduction to the Attacker Models for the IoT

Before the description and definition of the security assessment framework, we introduce an overview of the potential attacker models for the IoT at first. All

the attacker models presented in the thesis fall into this range. In specific, the attacker models for the IoT are classified by threats and discussed in [115]. We list them in the following.

- Eavesdropping/sniffing attacks: passive attackers can eavesdrop wireless or wired communication channels, capture packets and extract sensitive information from the packets. As many IoT devices have constrained resources and low computational power, it is infeasible to deploy strong security protections on the devices. For example, in a ZigBee-based network using standard security mode, the network key is transmitted over-the-air in plaintext [106, 139]. Therefore, eavesdropping can be easily carried out by malicious attackers and usually used as their first step for performing further attacks. Moreover, if there are internal attackers within the network, they are able to capture any information flow within the network.

- Denial of Service (DoS) attacks: in [141], a DoS attack is defined as any event that reduces or eliminates the network's normal functionality. Various factors can cause DoS attacks, including hardware failures, software vulnerabilities, resource exhaustion, *etc*. These attacks are more disruptive for the IoT as the IoT devices can be easily compromised due to limited security protections, and then controlled by attackers as zombie devices [147]. Moreover, such attacks have already been carried out by attackers in the real world [30].

- Physical damage: it is obvious that active attackers can destroy IoT devices easily as many devices are objects or properties located in our surrounding environment (e.g., home appliances, street lights, wearable devices).

- Node capture: instead of damaging the IoT devices, active attackers can also capture them, reprogram them and extract critical data from them [36]. For example, if a home-owner discards the ZigBee-based device without

the revocation of keys, the attacker is able to get the dumping device and launch the side-channel timing attack against the microcontroller by exploiting and programming JTAG [112].

- Controlling: if there is an attack path, active attackers might be able to manipulate the full or partial IoT network by compromising and gaining access to critical devices in the IoT.

3.2 Description and Formulation of the Framework

The main goals of the framework are to identify all possible attack paths in the IoT, evaluate the security level of the IoT through security metrics, and assess the effectiveness of different defence mechanisms. The proposed framework is shown in Figure 3.1.

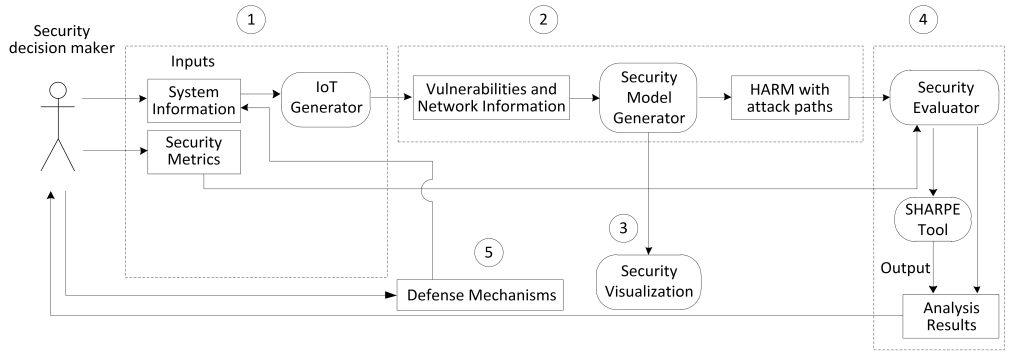


Figure 3.1: Phases in the proposed framework.

3.2.1 Framework Description

There are five phases in the framework: 1) data processing, 2) security model generation, 3) security visualisation, 4) security analysis, and 5) model updates. We explain each phase in the following.

In phase 1, the security decision maker (e.g., network security administrator) mainly provides two inputs needed to construct an IoT network: system

information and security metrics. First, the system information includes the subnets forming the IoT, node information and network topology for each subnet, and the vulnerability information for each node. The subnets can be classified based on the virtual local area networks (VLANs) that devices belong to, the communication protocols that devices use, *etc*. The classification method can be determined by the security decision maker. In this chapter, we use the communication protocols as the subnet classification method for the IoT nodes. Therefore some devices may belong to several subnets due to the communication protocols they use. The vulnerability information can be discovered by the scanning tools [91, 111, 120]. We collect and use the vulnerabilities from the Common Vulnerabilities and Exposures (CVE) [120] or existing literature. We use a static IoT network, thus all the inputs are fixed after the network generation. Then the system information is fed into the IoT Generator. The IoT Generator creates an IoT network consisting of the specified subnets with network topology information and node vulnerability information. Second, the security decision maker also selects the security metrics, which will be used as an input into the security analysis phase.

In phase 2, we generate the three-layered HARM of the IoT network created in phase 1. Specifically, the Security Model Generator takes the constructed network with topology and vulnerability information as inputs and generates the three-layered HARM of the network. Based on the HARM, we compute all possible attack paths in the IoT network. An attack path specifies a sequence of nodes that the attacker can compromise to reach the target node. The model supports multiple attackers and multiple targets (e.g., attackers in different places, a group of devices as the targets).

The three-layered HARM is based on the two-layered HARM developed in [59] and extends the HARM to three layers. They represent the subnet connectivity information in the upper layer, the network reachability information (i.e., nodes connected in the topological structure) in the middle layer and the vulnerability information of nodes in the lower layer, respectively. Compared to the two-

layered HARM, the three-layered HARM additionally describes the subnets and their connectivity. Apart from generating the three-layered HARM for the overall IoT, we can choose various sets of subnets, construct the HARM for the chosen set of subnets and also combine several three-layered HARMs based on the subnet connectivity. When the IoT contains a large number of nodes, the subnet division makes the model construction and further security evaluation more flexible. Additionally, the mobility of devices (e.g., node addition or removal) can be easily adjusted in the three-layer HARM without reconstruction of the whole model. Moreover, by grouping devices into different subnets based on their communication protocols, we are able to model any IoT with a wide variety of heterogeneous communication technologies, thus addressing the heterogeneity issue of the IoT. As each layer is constructed independently, the three-layered HARM improves the scalability of the two-layered HARM. Besides, the complexity of the security evaluation is further decreased because computations are grouped in each layer using a bottom-up approach. Lastly, more layers can be used based on different IoT scenarios. For example, a smart home with several networks (e.g., Wi-Fi, Bluetooth, *etc*) can be modelled using the three-layer HARM; a number of smart homes in an area can be modelled using the four-layer HARM with the home connectivity in the highest layer.

In phase 3, the HARM generated for the IoT network (including attack paths) is visualised in the form of an AG in the upper layer and middle layer, respectively, and a set of ATs in the lower layer.

In phase 4, the security analysis is carried out for the IoT network. The attack path information or other information (e.g., a set of nodes or vulnerabilities) is taken as an input into the Security Evaluator along with the determined security metrics. Based on the metrics, the Security Evaluator can perform one of two options. One option is to output the analysis results directly and the other option is to generate a textual input file and export the file into the SHARPE which computes the security analysis results. The description and calculation of security metrics are presented in Section 3.2.2.

In specific, the SHARPE [121] is a software package for performance and reliability analysis of computer systems. It accepts a mathematical model of the system and analyses it using various algorithms. Several model types are provided, for example, Markov chains, Semi-Markov chains, reliability block diagrams, fault trees and reliability graphs. Each model type supports at least one analysis algorithm; for example, fault trees have five analysis algorithms including reliability, unreliability, mean-time-to-failure, *etc*. Given the behaviour of the components of a system in the form of time-dependent functions and the structure of the system in the form of a model type, the SHARPE can compute the behaviour of the system as a function of time which is used for performance and reliability analysis.

In phase 5, any changes caused by the defence mechanisms are captured to update model inputs. Based on the security analysis results, the security decision maker knows which part of the IoT is the most vulnerable, thus being able to decide proper defence mechanisms. The deployment of a defence mechanism changes either the vulnerability information (e.g., eliminates a specific vulnerability in an IoT node or mitigates the effect caused by the vulnerability) or the topology information (e.g., IoT node removal or addition), which is updated and taken as the input into the IoT Generator. The previous phases are carried out again to re-analyse the security of the network after the deployment of the defence mechanism. When choosing the defence mechanisms, the security decision maker can also assess the effectiveness of different mechanisms via the framework by using security metrics and comparing their effects.

3.2.2 Framework Formulation

We formally describe the framework in terms of the network, subnet, node and vulnerability information, and then define the three-layered HARM.

3.2.2.1 General Notations

An IoT network has three major attributes, which are a finite set of subnets S, a finite set of IoT nodes T, and a finite set of vulnerabilities V. We denote a subnet as $s \in S$, a node as $t \in T$ and a vulnerability as $v \in V$. For one target, the attacker may be able to find multiple attack paths to reach it via one or multiple entry points. Thus we consider a set of all attack paths AP for reaching a given target or multiple targets. Each attack path $ap \in AP$ is a sequence of nodes and each node in the path has one or more vulnerabilities. The notations and definitions of security metrics used in the framework are listed in Table 3.1.

We also define two metrics for analysing the node and network properties in the following. These metrics will be used to analyse the impact of a routing attack in the smart home scenario in Section 3.3.1.

- Node degree (d_t): degree of a single node t
- Average node connectivity (ANC): average of local node connectivity over all pairs of nodes in the network

The attributes of an IoT network $IoT = (S, T, V)$ are shown as follows:

- Each subnet $s \in S$ has a name s_{name}, a set of IoT nodes $s_{nodes} \subseteq T$, a topology information $s_{topo} \in \{tree, mesh, ...\}$, and a set of adjacent subnets $s_{adj} \subseteq S$ according to the network structure.
- Each node $t \in T$ has a name t_{name}, a type $t_{type} \in \{Smart\ TV\ (Samsung), Tablet\ (Android), ...\}$, a mobility information $t_{mobility} \in \{static, mobile\}$, a set of adjacent nodes $t_{adj} \subseteq T$ according to the network structure, a set of vulnerabilities $t_{vuls} \subseteq V$, and a set of security metrics $t_{metrics} \subseteq \{d_t, asp_t, ac_t, aim_t, mttc_t, r_t, roa_t\}$.
- Each vulnerability $v \in V$ has a name v_{name}, a privilege level that is acquired by the attacker after the vulnerability is successfully exploited

Table 3.1: Notations and definitions of security metrics.

Metrics	Notations	Definitions
Vulnerability level		
• Attack success probability	asp_v	Probability of an attacker to successfully exploit a vulnerability ([0, 1])
• Attack cost	ac_v	Cost spent by an attacker who successfully exploits a vulnerability ([0, 10])
• Attack impact	aim_v	Potential loss caused by an attacker who successfully exploits a vulnerability ([0, 10])
• Compromise rate	cr_v	Frequency of the attacker to successfully exploit a vulnerability per unit of time (one hour)
• Risk	r_v	Potential harm caused by an attacker who successfully exploits a vulnerability
• Return-on-attacks	roa_v	Potential gain on an attacker's effort to successfully exploit a vulnerability
Node level		
• Attack success probability	asp_t	Probability of an attacker to successfully compromise a node
• Attack cost	ac_t	Minimum cost spent by an attacker who successfully compromises a node
• Attack impact	aim_t	Maximum potential loss caused by an attacker who successfully compromises a node
• Mean-time-to-compromise	$mttc_t$	Mean time for the attacker to successfully compromise a node
• Risk	r_t	Potential harm caused by an attacker who successfully compromises a node
• Return-on-attacks	roa_t	Potential gain on an attacker's effort to successfully compromise a node
Attack path level		
• Attack success probability	asp_{ap}	Probability of an attacker to successfully compromise the target via an attack path
• Attack cost	ac_{ap}	Cost spent by an attacker who successfully compromises the target via an attack path
• Attack impact	aim_{ap}	Potential loss caused by an attacker who successfully compromises the target via an attack path
• Mean-time-to-compromise	$mttc_{ap}$	Mean time for the attacker to successfully compromise the target via an attack path
• Risk	r_{ap}	Potential harm caused by an attacker who successfully compromises the target via an attack path
• Return-on-attacks	roa_{ap}	Potential gain on an attacker's effort to successfully compromise the target via an attack path
Network level		
• Attack success probability	ASP	Probability of an attacker to successfully compromise the target via all potential attack paths
• Attack cost	AC	Minimum cost spent by an attacker who successfully compromises the target
• Attack impact	AIM	Maximum potential loss caused by an attacker who successfully compromises the target
• Mean-time-to-compromise	$MTTC$	Mean time for the attacker to successfully compromise the target in minimal attack sequences
• Risk	R	Maximum potential harm caused by an attacker who successfully compromises the target
• Return-on-attacks	ROA	Maximum potential gain on an attacker's effort to successfully compromise the target
• Number of attack paths	NAP	Potential paths for an attacker to reach the target
• Mean-attack-path-length	$MAPL$	Expected effort for an attacker to reach the target

$v_{privilege} \in \{root, user, ...\}$, and a set of security metrics $v_{metrics} \subseteq \{asp_v, ac_v, aim_v, cr_v, r_v, roa_v\}$.

3.2.2.2 Security Model Definition

We define the three-layered HARM based on the HARM [59]. The three-layered HARM has upper, middle and lower layers. The upper layer model (an AG) represents the subnet connectivity information and the attackers' entry points, the middle layer model (an AG) captures the network reachability information and the attackers' entry points, and the lower layer model (a set of ATs) depicts the vulnerability information of each node (if the node has vulnerabilities) and an attack goal achieved by the attackers by exploiting one or multiple vulnerabilities.

Definition 1. The three-layered HARM of an IoT network $IoT = (S, T, V)$ is defined as a 5-tuple $GSM = (U, M, L, C_{U,M}, C_{M,L})$. Here, U is an AG model for S (the upper layer), M is an AG model for T (the middle layer) and L is a set of AT models for V (the lower layer). The relationship between components in the upper and middle layers is described by $C_{U,M} = \{(s, t) \mid s \in S \text{ and } t \in s_{nodes}\} \subseteq S \times T$. Each node that has one or more vulnerabilities has a corresponding AT in the lower layer; the partial mapping $C_{M,L} : T \rightarrow L$ gives the associated AT.

Definition 2. An AG is defined as a directed graph $ag = (N, E)$ where N is a finite set of components and $E \subseteq N \times N$ is a set of edges between components. Let k be the subnet including one or multiple attackers where $k \notin S$ and $k_{nodes} \cap T = \emptyset$. The representations of U and M are as follows:

- U: $N \subseteq S \cup \{k\}$ and $E \subseteq (S \cup \{k\}) \times S$
- M: $N \subseteq T \cup k_{nodes}$ and $E \subseteq (T \cup k_{nodes}) \times T$.

The restrictions on the edges imply that there are no edges into the attacker subnet or its nodes.

Definition 3. An AT is defined as a 5-tuple $at = (A, B, c, g, root)$. Here, A is

a set of components which are the leaves of at and B is a set of gates which are the inner nodes of at. We require $A \cap B = \emptyset$ and $root \in A \cup B$. Let $\mathscr{P}(X)$ denote the power set of X. The function $c : B \rightarrow \mathscr{P}(A \cup B)$ describes the children of each inner node in at (we assume there are no cycles). The function $g : B \rightarrow \{AND, OR\}$ describes the type of each gate. The representation of the attack tree at_t associated to the node $t \in T$ is as follows:

- at_t: $A \subseteq t_{vuls}$.

This means that the vulnerabilities of a node are combined using logical *AND* and *OR* gates.

3.2.2.3 Calculation of Security Metrics

The security metrics, shown in Table 3.1, are divided into four levels, which are the network, attack path, node and vulnerability levels. The values of some metrics in higher levels are calculated from lower levels in the security analysis phase. This is done for attack success probability, attack cost, attack impact, mean-time-to-compromise, risk and return-on-attacks [28, 84]. For example, values in the network level are calculated from values in the attack path, node and vulnerability levels. The value of the attack success probability is in the range of zero to one, while the value of ac_v and aim_v is in the range of zero to ten. Take the attack success probability as an example. The larger the value is within the range, the higher the probability is for an attacker to exploit the vulnerability. By introducing the value range, we use standardised metric values as it is not easy to get the exact values of the security metrics from the real-world scenarios. The Common Vulnerability Scoring System (CVSS) [45] uses a similar way to assess the severity of vulnerabilities.

We calculate the attack cost, attack impact, risk, return-on-attacks and mean-attack-path-length using the Security Evaluator. For the attack success probability and mean-time-to-compromise, we use the Security Evaluator and the SHARPE. In the following calculations, for each node $t \in T$ that has an attack tree

$at_t = (A, B, c, g, root)$, we assign values to asp_v, ac_v, aim_v and cr_v for each $v \in A$ based on the CVSS and existing research papers that analyse the vulnerabilities.

Attack success probability: attack success probability is used to measure the probability of an attacker to successfully achieve an attack goal. At the node level, the metric is the probability for an attacker to compromise the node. At first, we calculate the attack success probability values for each inner node of an attack tree by Equation (3.1). Then the attack success probability value of a node $t \in T$ is the attack success probability value of the root of the corresponding attack tree by Equation (3.2). At the path level, the metric is the probability for an attacker to compromise the target via the attack path. The attack success probability value of an attack path is calculated by Equation (3.3). At the network level, the metric is the probability for an attacker to compromise the target via all potential paths. The network-level value *ASP* is calculated in Algorithm 1.

$$asp_b = \begin{cases} \prod\limits_{a \in c(b)} asp_a, & \substack{b \in B \\ g(b)=AND} \\ 1 - \prod\limits_{a \in c(b)} (1 - asp_a), & \substack{b \in B \\ g(b)=OR} \end{cases} \tag{3.1}$$

$$asp_t = asp_{root} \tag{3.2}$$

$$asp_{ap} = \prod_{t \in ap} asp_t, \quad ap \in AP \tag{3.3}$$

In Algorithm 1, we use the reliability graph model in the SHARPE to calculate the probability that there is no attack path from the attacker to the target and then use 1 minus that probability to calculate *ASP*. Specifically, the SHARPE analyses the reliability graph using the factoring algorithm [121]. After factoring, if the sub-graph becomes series-parallel, its analysis can be done using Equation (3.4) where F is the distribution function of time variable i and J is the number of

Algorithm 1: Calculation of *ASP*

Data: AP and asp_t ($t \in ap$)
Result: ASP

```
begin
    H ← {t | t∈ap for some ap∈AP}
    Construct a directed graph graph with node set H
    for each attack path (t_1, ..., t_n)∈AP do
        for each i∈{2, ..., n} do
            Include edge (t_{i-1}, t_i) with value 1 − asp_{t_i} in graph
        end
    end
    ASP ← 1 − CalculateProbability(graph)
end
```

nodes included in the structure.

$$F(i) = \begin{cases} 1 - \prod_{j=1}^{J}[1 - F_j(i)], & \text{for a series structure} \\ \prod_{j=1}^{J} F_j(i), & \text{for a parallel structure} \end{cases} \tag{3.4}$$

Attack cost: attack cost is used to measure the cost spent by an attacker to successfully achieve an attack goal. At the node level, the metric is the cost spent by an attacker to compromise a node. Attack cost values for each inner node of an attack tree and each node $t \in T$ are calculated by Equations (3.5) and (3.6). At the path level, the metric is the cost spent by an attacker to compromise the target via the attack path. The attack cost value of an attack path is calculated by Equation (3.7). At the network level, the metric is the minimum cost spent by an attacker to compromise the target among all potential paths. The network-level value AC is thus given by Equation (3.8).

$$ac_b = \begin{cases} \sum\limits_{a \in c(b)} ac_a, & \substack{b \in B \\ g(b) = AND} \\ \min\limits_{a \in c(b)} ac_a, & \substack{b \in B \\ g(b) = OR} \end{cases} \tag{3.5}$$

$$ac_t = ac_{root} \tag{3.6}$$

$$ac_{ap} = \sum_{t \in ap} ac_t, \quad ap \in AP \tag{3.7}$$

$$AC = \min_{ap \in AP} ac_{ap} \tag{3.8}$$

Attack impact: attack impact is used to compute the potential loss caused by an attacker to successfully achieve an attack goal. The potential loss is the loss of confidentiality, integrity and availability. At the node level, the metric is the loss caused by an attacker to compromise a node. Attack impact values for each inner node of an attack tree and each node $t \in T$ are calculated by Equations (3.9) and (3.10). At the path level, the metric is the loss caused by an attacker to compromise the target via the attack path. The attack impact value of an attack path is calculated by Equation (3.11). In the network level, the metric is the maximum loss caused by an attacker to compromise the target among all potential paths. The network-level value *AIM* is thus given by (3.12).

$$aim_b = \begin{cases} \sum\limits_{a \in c(b)} aim_a, & \substack{b \in B \\ g(b) = AND} \\ \max\limits_{a \in c(b)} aim_a, & \substack{b \in B \\ g(b) = OR} \end{cases} \tag{3.9}$$

$$aim_t = aim_{root} \tag{3.10}$$

$$aim_{ap} = \sum_{t \in ap} aim_t, \quad ap \in AP \tag{3.11}$$

$$AIM = \max_{ap \in AP} aim_{ap} \tag{3.12}$$

Mean-time-to-compromise: mean-time-to-compromise is used to measure the mean time for an attacker to successfully achieve an attack goal. At the node

level, the metric is the mean time for an attacker to compromise a node. If the node has only one vulnerability, which means the AT contains just one node with a compromise rate cr_{root}, we obtain the mean-time-to-compromise value $mttc_t$ by Equation (3.13). If the node has more than one vulnerability, which means the AT has more than one node, we use Algorithm 2.

$$mttc_t = 1/cr_{root} \tag{3.13}$$

Algorithm 2: Calculation of $mttc_t$

Data: at_t and cr_v ($v \in A$)

Result: $mttc_t$

```
begin
    Create a tree tree with structure at_t and values cr_v as leaves
    mttc_t ← CalculateMean(tree)
end
```

In Algorithm 2, we use the fault tree model in the SHARPE to calculate the mean-time-to-compromise. Specifically, the SHARPE analyses the fault tree with repeated components using the factoring algorithm [121]. After factoring, if the sub-tree has no repeated components, its analysis can be done using Equation (3.14) where F is the distribution function of time variable i and J is the number of nodes included in the structure.

$$F(i) = \begin{cases} \prod_{j=1}^{J} F_j(i), & \text{for } AND \text{ gate} \\ 1 - \prod_{j=1}^{J} [1 - F_j(i)], & \text{for } OR \text{ gate} \end{cases} \tag{3.14}$$

At the path level, the metric is the mean time for an attacker to compromise the target via the attack path. We calculate $mttc_{ap}$ by Equation (3.15). At the network level, the metric is the minimum mean time for an attacker to compromise the target among all potential attack paths. We calculate *MTTC* by Equation (3.16).

$$mttc_{ap} = \sum_{t \in ap} mttc_t, \quad ap \in AP \tag{3.15}$$

$$MTTC = \min_{ap \in AP} mttc_{ap} \tag{3.16}$$

Risk: risk is used to compute the potential harm caused by an attacker to successfully achieve an attack goal. At the vulnerability level, the metric is the harm caused by an attacker to exploit a vulnerability and calculated from the attack success probability value times the attack impact value. For each vulnerability $v \in A$, the risk value is calculated by Equation (3.17). At the node level, the metric is the harm caused by an attacker to compromise a node. Risk values for each inner node of an attack tree and each node $t \in T$ are calculated by Equations (3.18) and (3.19). At the path level, the metric is the harm caused by an attacker to compromise the target via the attack path. The risk value of an attack path is calculated by Equation (3.20). In the network level, the metric is the maximum harm caused by an attacker to compromise the target among all potential paths. The network-level value R is thus given by (3.21).

$$r_v = asp_v * aim_v, \quad v \in A \tag{3.17}$$

$$r_b = \begin{cases} \sum_{a \in c(b)} r_a, & \substack{b \in B \\ g(b)=AND} \\ \max_{a \in c(b)} r_a, & \substack{b \in B \\ g(b)=OR} \end{cases} \tag{3.18}$$

$$r_t = r_{root} \tag{3.19}$$

$$r_{ap} = \sum_{t \in ap} r_t, \quad ap \in AP \tag{3.20}$$

$$R = \max_{ap \in AP} r_{ap} \tag{3.21}$$

Return-on-attacks: return-on-attacks is used to compute the potential gain on an attacker's effort to successfully achieve an attack goal. At the vulnerability

level, the metric is the gain on an attacker's effort to exploit a vulnerability and calculated from the risk value divided by the attack cost value. For each vulnerability $v \in A$, the return-on-attacks value is calculated by Equation (3.22). At the node level, the metric is the gain on an attacker's effort to compromise a node. Return-on-attacks values for each inner node of an attack tree and each node $t \in T$ are calculated by Equations (3.23) and (3.24). At the path level, the metric is the gain on an attacker's effort to compromise the target via the attack path. The return-on-attacks value of an attack path is calculated by Equation (3.25). In the network level, the metric is the maximum gain on an attacker's effort to compromise the target among all potential paths. The network-level value *ROA* is thus given by (3.26).

$$roa_v = r_v / ac_v, \quad v \in A \tag{3.22}$$

$$roa_b = \begin{cases} \sum\limits_{a \in c(b)} roa_a, & \substack{b \in B \\ g(b)=AND} \\ \max\limits_{a \in c(b)} roa_a, & \substack{b \in B \\ g(b)=OR} \end{cases} \tag{3.23}$$

$$roa_t = roa_{root} \tag{3.24}$$

$$roa_{ap} = \sum_{t \in ap} roa_t, \quad ap \in AP \tag{3.25}$$

$$ROA = \max_{ap \in AP} roa_{ap} \tag{3.26}$$

Number of attack paths: number of attack paths is used to compute the potential paths for an attacker to successfully reach the target and calculated in the network level (i.e., the middle layer of the three-layered HARM) by Equation (3.27).

$$NAP = |AP| \tag{3.27}$$

Mean-attack-path-length: mean-attack-path-length is used to compute the expected effort for an attacker to successfully reach the target and calculated in the network level (i.e., the middle layer of the three-layered HARM). The mean-attack-path-length value is calculated by Equation (3.28).

$$MAPL = \frac{\sum_{ap \in AP} |ap|}{|AP|} \tag{3.28}$$

3.3 Evaluation of the Framework

The IoT has been widely applied in various fields, including healthcare, transport, environment monitoring, *etc*. We use three example networks in three different scenarios to show the feasibility and scalability of the framework. They are the home network in a smart home, the wireless body area network in the wearable healthcare monitoring scenario and the wireless sensor network for environment monitoring. Any example networks in other scenarios can also be used to evaluate the framework.

3.3.1 Sinkhole Attack in Smart Home

A smart home is one of the application domains of the emerging IoT. It has come into thousands of families and brought new technologies to people's lives [53]. Unfortunately, it also provides a platform for attackers to hack into the home network and remotely control home systems. As many IoT devices are resource-constrained, standard security solutions may not be implemented. IoT devices can become entry points into the smart home and can then be exploited to leak sensitive information [145]. Thus the smart home environment is exposed to various threats. With an increasing number of Internet-connected devices in the house, vulnerabilities and related threats also increase. We describe the attack scenarios in the smart home and show the benefits of the framework via a use case.

3.3.1.1 Scenario Description

A smart home is formed by a number of home automation systems, which can autonomously operate devices and thus control the home on behalf of users [67].

ZigBee technology, an IEEE 802.15.4-based specification [72], is designed to be used by applications that require low data rate, low cost, low power consumption and two-way wireless communications. Some examples are home appliances (e.g., air conditioners, refrigerators, and washing machines), lighting control (e.g., light bulbs), environment monitoring (e.g., temperature, humidity) and security (e.g., smart door lock, surveillance camera).

The Wi-Fi standard has been widely established as the wireless home networking technology. It is designed to provide relatively high data rate communications. It can be used for multimedia applications of digital products in the home network (smartphones, smart TVs, tablets, *etc*).

3.3.1.2 System Model

We consider the IoT-enabled home network shown in Figure 3.2 as an example. The home network is a heterogeneous network with devices using different operating systems, applications and communication protocols. It includes a ZigBee network and a Wi-Fi network. As ZigBee and Wi-Fi can coexist with less interference problems than alternative technologies, the combination of them has the potential to provide comprehensive home network solutions [51]. A smart home automation hub is used to support Wi-Fi, ZigBee and Internet communications. Specifically, the integrated hub is able to establish a ZigBee network that allows home devices to communicate with each other by using the ZigBee wireless protocol; it provides the Internet connection for the ZigBee network and the Wi-Fi network; it also provides a user interface control panel so that users can connect to the hub through the Internet to get access to ZigBee devices and remotely control them [34].

The ZigBee network contains heterogeneous sensors [37, 71]. Our use case

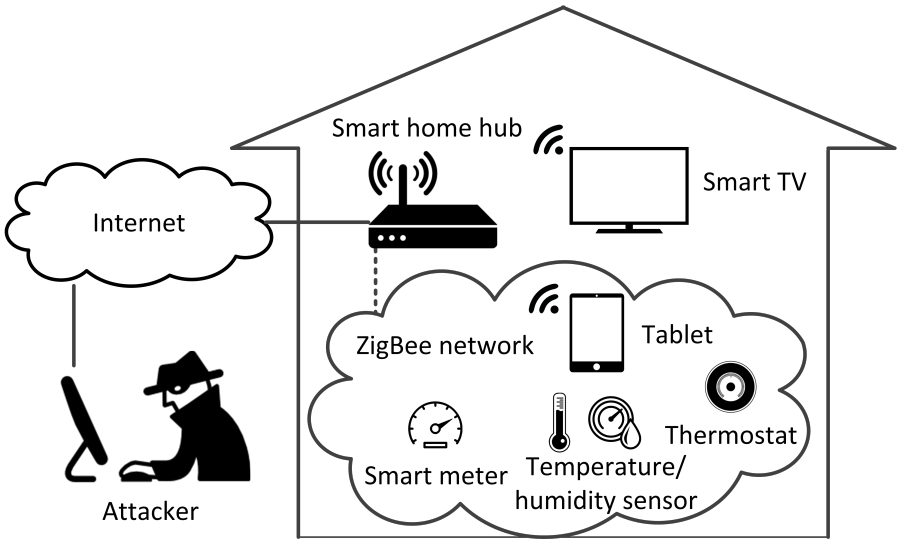


Figure 3.2: A smart home scenario.

has a number of ZigBee devices presented in the emulated environment in [37], such as electricity meters, thermostats, temperature and humidity measurement sensors. ZigBee devices communicate wirelessly to the hub (acting as the coordinator) in the form of a mesh topology. Some devices act as routers to extend the limited range of the network (e.g., the electricity meter). They can transfer packets to/from other ZigBee devices. Some devices are end devices thus only interacting with a router or the hub (e.g., thermostat, temperature and humidity measurement sensor).

We use an Android tablet equipped with a ZigBee chip. It connects to both the Wi-Fi network and the ZigBee network and acts as a ZigBee router in the ZigBee network. We also use a Smart TV which connects to the Wi-Fi network. Both of them get access to the Internet through the smart home hub.

3.3.1.3 Attacker Model

We assume the attacker's goal is to lure the traffic from the smart home hub through a compromised device as the ZigBee routing algorithm is prone to Sinkhole attacks [37]. The assumptions about the attacker's ability are listed

in the following.

1. The attacker is able to remotely compromise the Smart TV. In the literature, there are several papers addressing remote attacks on smart TVs [20, 98]. According to the practical proof-of-concept attacks or experiments introduced in the papers, the attackers can remotely exploit software vulnerabilities on a smart TV without physical proximity to the target and gain control over it. Then they can use it as a gateway to exploit vulnerabilities in any other devices inside the home (e.g., the tablet).

2. The attacker is able to compromise the Android tablet with a specific malware exploiting several bugs in the software and the operating system [37]. After the tablet is compromised, the attacker can use the tablet to launch other attacks targeting the ZigBee network. We use the Sinkhole attack as an example of a further attack as the ZigBee routing algorithm is prone to the Sinkhole attack. The attacker is able to use a malicious device (i.e., the Android tablet) to advertise false routing table stating a shorter route to the hub, thus luring traffic of some ZigBee home devices from the hub to the malicious device.

Specifically, we use the attacks introduced in [37, 98]. For the smart TV, the attacker can construct a malicious media file by using FFmpeg to find exploitable vulnerabilities in supported media formats and upload the file on the Internet. After the victim downloads the malicious file and starts to play back the video file, the TV is compromised. We assume FFmpeg 5.0 is used by the TV. Two vulnerabilities are found in two types of media file formats supported by FFmpeg and the attacker can exploit any one of them to run arbitrary code and gain the root privilege of the TV. The information about the two vulnerabilities in the CVE and their CVSS base scores are summarised in Table 3.2. For the Android tablet, the attacker can write a malware to get the root permission and change the transmission power of the ZigBee chip integrated in the device. The malware was developed based on a malfunctional Trojan,

Backdoor.AndroidOS.Obad.a. According to [138], it exploits three bugs: firstly, an error in the DEX2JAR software was used to disrupt the conversion of Dalvik bytecode into Java bytecode, which complicates the statistical analysis of the Trojan; secondly, an error in the Android operating system was used to modify the AndroidManifest.xml file, which makes a dynamic analysis of the Trojan extremely hard; thirdly, a previously unknown error in the Android operating system was used to obtain the extended Device Administrator privileges without appearing on the list of applications which have such privileges, which makes the detection impossible.

Table 3.2: Vulnerability information in the TV.

CVE ID	CVSS Base score	Impact	Exploitability
CVE-2008-4866	10.0	10.0	10.0
CVE-2009-0385	9.3	10.0	8.6

3.3.1.4 Data Processing

As the devices in the home network use Wi-Fi and ZigBee communication protocols, we introduce two subnets to differentiate heterogeneous devices based on our classification method. The subnets are denoted as s_{wifi} and s_{zigbee} respectively. In the ZigBee network, we use 5 devices acting as routers and 3 end devices attached to each router. ZigBee routers are denoted as t_{ri} ($i \in \{1,2,...,5\}$) and ZigBee end devices are denoted as t_{ej} ($j \in \{1,2,...,15\}$). The smart home hub denoted as t_{hub} and an Android tablet denoted as t_{tab} belong to both ZigBee network and Wi-Fi network. The Wi-Fi network also includes a TV denoted as t_{tv}. In the IoT Generator, the IoT network is represented as $IoT_1 = (S_1, T_1, V_1)$ where S_1=$\{s_{wifi}, s_{zigbee}\}$, T_1=$\{t_{hub}, t_{tv}, t_{tab}, t_{r1}, ..., t_{r5}, t_{e1}, ..., t_{e15}\}$ and V_1=$\{v_{tv1}, v_{tv2}, v_{tab1}, v_{tab2}, v_{tab3}\}$. Here, v_{tv1} and v_{tv2} refer to two vulnerabilities found in two types of media file formats in the TV, namely CVE-2008-4866 and CVE-2009-0385 in Table 3.2. Three software bugs in the tablet are denoted as v_{tab1}, v_{tab2} and v_{tab3}.

We show the full list of attributes for a subnet s_{zigbee}, a node t_{tv} and a vulnerability v_{tv1} as examples in the following.

- $s_{zigbee_{name}} = zigbee$
- $s_{zigbee_{nodes}} = \{t_{tab}, t_{r1}, ..., t_{r5}, t_{e1}, ..., t_{e15}\}$
- $s_{zigbee_{topo}} = mesh$
- $s_{zigbee_{adj}} = \{s_{wifi}\}$
- $t_{tv_{name}} = tv$
- $t_{tv_{type}} = Smart\ TV\ (Samsung)$
- $t_{tv_{mobility}} = static$
- $t_{tv_{adj}} = \{t_{hub}, t_{tab}\}$
- $t_{tv_{vuls}} = \{v_{tv1}, v_{tv2}\}$
- $t_{tv_{metrics}} = \{d_{t_{tv}}, asp_{t_{tv}}, ac_{t_{tv}}, aim_{t_{tv}}, mttc_{t_{tv}}, r_{t_{tv}}, roa_{t_{tv}}\}$
- $v_{tv1_{name}} = CVE\text{-}2008\text{-}4866$
- $v_{tv1_{privilege}} = root$
- $v_{tv1_{metrics}} = \{asp_{v_{tv1}}, ac_{v_{tv1}}, aim_{v_{tv1}}, cr_{v_{tv1}}, r_{v_{tv1}}, roa_{v_{tv1}}\}$

We choose the following security metrics to be used in the security analysis phase: attack success probability, attack cost, attack impact, mean-time-to-compromise, risk and return-on-attacks. Based on the vulnerabilities described in Section 3.3.1.3, we make assumptions about the metric values of vulnerabilities in the TV and the Android tablet and show the values in Table 3.3. For the values of vulnerabilities in the TV, we extract the values of attack impact and attack success probability from the impact and exploitability scores in Table 3.2 respectively, and estimate the values of the other two security metrics from the CVSS base scores. Both v_{tv1} and v_{tv2} allow an attacker to exploit from the Internet

without any authentication. Both vulnerabilities do not require special tools by the attacker. A laptop-class device with a Linux-based operating system and hacking software would be sufficient. Therefore we use low attack cost for v_{tv1} and v_{tv2}. The compromise rate indicates the frequency that the vulnerability can be exploited successfully. We estimate the compromise rate as once per week as the victim may download the video files at weekends and accidentally get a malicious one.

For the metric values of vulnerabilities in the Android tablet, we can estimate the values based on the descriptions as no CVSS scores are available. All three vulnerabilities in the tablet allow an attacker to exploit from the Internet without any authentication. There is no need for special tools used by the attacker to exploit vulnerabilities. We assume low access complexity as the attacker can easily gain the knowledge of the Android operating system. Thus we use low attack cost and high attack success probability. As people might use their tablets every day and accidentally download the malware, we assume the compromise rate as twice per week. Vulnerabilities v_{tab1} and v_{tab2} can be exploited to complicate the analysis of the Trojan, which are assumed to have low attack impact (i.e., partial impact on confidentiality, integrity and availability). Vulnerability v_{tab3} is used to obtain the extended privileges which is assumed to have the maximum attack impact.

Table 3.3: Metric values for each vulnerability in the home network.

Metric / Vulnerability	asp_v	ac_v	aim_v	cr_v
v_{tv1}	1.0	3.0	10.0	0.006
v_{tv2}	0.86	3.0	10.0	0.006
v_{tab1}	0.8	3.0	4.0	0.012
v_{tab2}	0.8	3.0	4.0	0.012
v_{tab3}	0.8	3.0	10.0	0.012

3.3.1.5 Security Model Generation and Visualisation

We use $IoT_1 = (S_1, T_1, V_1)$ as the input into the Security Model Generator and compute the three-layered HARM. The model is represented as $GSM_1 = (U_1, M_1, L_1, C_{U_1,M_1}, C_{M_1,L_1})$.

As an example of an attack graph, we show U_1=($\{k, s_{zigbee}, s_{wifi}\}$, $\{k \rightarrow s_{wifi}, s_{wifi} \rightarrow s_{zigbee}\}$).

As an example of an attack tree, we show $at_{t_{tv}}$=($\{v_{tv1}, v_{tv2}\}$, $\{root_{tv}\}$, $c(root_{tv})$=$\{v_{tv1}, v_{tv2}\}$, $g(root_{tv})$=OR, $root_{tv}$).

We use two subnets in the upper layer of the model to represent s_{zigbee} and s_{wifi}. Figure 3.3 shows the visualised attack path in the network captured by the model. By exploiting the vulnerabilities, the attacker is able to bypass the smart home hub and break into the home network via the smart TV.

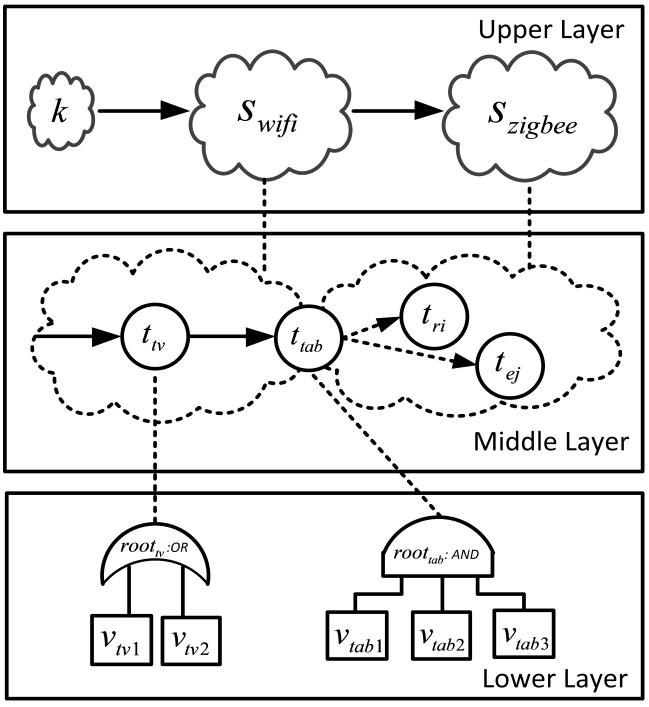


Figure 3.3: Attack paths in the home network.

3.3.1.6 Security Analysis and Model Updates

In the Sinkhole attack, more devices choose the malicious tablet to route their data to the smart home hub as the compromised tablet represents a shorter route to the hub with the increased power and increased probability of successfully delivering the packets [37]. We assume the compromised tablet refuses to deliver any packets to/from the hub. Thus we analyse the impact of the attack using the average node connectivity of the network (ANC_1) and the degree of the hub and the tablet ($d_{t_{hub}}$ and $d_{t_{tab}}$) shown in Table 3.4. In our example network, under the Sinkhole attack, the average node connectivity drops as the malicious tablet disconnects with the hub and the TV which partitions the network into two separate parts; the degree of the hub decreases while the degree of the tablet increases as some routers and end devices cut off their initial connections and connect to the tablet because of its higher transmission power.

Table 3.4: Impact of the Sinkhole attack on the network connectivity.

Metric / Situation	ANC_1	$d_{t_{hub}}$	$d_{t_{tab}}$
Before the attack	1.1146	4	2
After the attack	1.1028	3	10

As there is only one attack path ap=(t_{tv}, t_{tab}) in the network, we calculate values of security metrics in the node and attack path levels. **Attack success probability:** we calculate $asp_{t_{tv}}$ and $asp_{t_{tab}}$ by Equations (3.1) and (3.2), asp_{ap} by Equation (3.3).

$$\begin{aligned} asp_{t_{tv}} &= asp_{root_{tv}} = 1-(1-asp_{v_{tv1}})*(1-asp_{v_{tv2}}) \\ &= 1.0-(1.0-1.0)*(1.0-0.86) = 1.0 \\ asp_{t_{tab}} &= asp_{root_{tab}} = asp_{v_{tab1}}*asp_{v_{tab2}}*asp_{v_{tab3}} \\ &= 0.8*0.8*0.8 = 0.512 \\ asp_{ap} &= asp_{t_{tv}}*asp_{t_{tab}} = 1.0*0.512 = 0.512 \end{aligned}$$

Attack cost: we calculate $ac_{t_{tv}}$ and $ac_{t_{tab}}$ by Equations (3.5) and (3.6), ac_{ap} by Equation (3.7).

$$\begin{aligned} ac_{t_{tv}} &= ac_{root_{tv}} = min(ac_{v_{tv1}}, ac_{v_{tv2}}) \\ &= min(3.0, 3.0) = 3.0 \\ ac_{t_{tab}} &= ac_{root_{tab}} = ac_{v_{tab1}} + ac_{v_{tab2}} + ac_{v_{tab3}} \\ &= 3.0 + 3.0 + 3.0 = 9.0 \\ ac_{ap} &= ac_{t_{tv}} + ac_{t_{tab}} = 3.0 + 9.0 = 12.0 \end{aligned}$$

Attack impact: we calculate $aim_{t_{tv}}$ and $aim_{t_{tab}}$ by Equations (3.9) and (3.10), aim_{ap} by Equation (3.11).

$$\begin{aligned} aim_{t_{tv}} &= aim_{root_{tv}} = max(aim_{v_{tv1}}, aim_{v_{tv2}}) \\ &= max(10.0, 10.0) = 10.0 \\ aim_{t_{tab}} &= aim_{root_{tab}} = aim_{v_{tab1}} + aim_{v_{tab2}} + aim_{v_{tab3}} \\ &= 4.0 + 4.0 + 10.0 = 18.0 \\ aim_{ap} &= aim_{t_{tv}} + aim_{t_{tab}} = 10.0 + 18.0 = 28.0 \end{aligned}$$

Mean-time-to-compromise: we use Algorithm 2 to calculate $mttc_{t_{tv}}$ and $mttc_{t_{tab}}$. The SHARPE outputs are shown in the following: $mttc_{t_{tv}}$ = 83.33 and $mttc_{t_{tab}}$ = 152.78. We also calculate $mttc_{ap}$ by Equation (3.15).

$$mttc_{ap} = mttc_{t_{tv}} + mttc_{t_{tab}} = 83.33 + 152.78 \approx 236.11$$

Risk: we calculate $r_{v_{tv1}}$, $r_{v_{tv2}}$, $r_{v_{tab1}}$, $r_{v_{tab2}}$ and $r_{v_{tab3}}$ by Equation (3.17), $r_{t_{tv}}$ and

$r_{t_{tab}}$ by Equations (3.18) and (3.19), r_{ap} by Equation (3.20).

$$
\begin{aligned}
r_{v_{tv1}} &= asp_{v_{tv1}} * aim_{v_{tv1}} = 1.0 * 10.0 = 10.0 \\
r_{v_{tv2}} &= asp_{v_{tv2}} * aim_{v_{tv2}} = 0.86 * 10.0 = 8.6 \\
r_{v_{tab1}} &= asp_{v_{tab1}} * aim_{v_{tab1}} = 0.8 * 4.0 = 3.2 \\
r_{v_{tab2}} &= asp_{v_{tab2}} * aim_{v_{tab2}} = 0.8 * 4.0 = 3.2 \\
r_{v_{tab3}} &= asp_{v_{tab3}} * aim_{v_{tab3}} = 0.8 * 10.0 = 8.0 \\
r_{t_{tv}} &= r_{root_{tv}} = max(r_{v_{tv1}}, r_{v_{tv2}}) = max(10.0, 8.6) = 10.0 \\
r_{t_{tab}} &= r_{root_{tab}} = r_{v_{tab1}} + r_{v_{tab2}} + r_{v_{tab3}} = 3.2 + 3.2 + 8.0 = 14.4 \\
r_{ap} &= r_{t_{tv}} + r_{t_{tab}} = 10.0 + 14.4 = 24.4
\end{aligned}
$$

Return-on-attacks: we calculate $roa_{v_{tv1}}$, $roa_{v_{tv2}}$, $roa_{v_{tab1}}$, $roa_{v_{tab2}}$ and $roa_{v_{tab3}}$ by Equation (3.22), $roa_{t_{tv}}$ and $roa_{t_{tab}}$ by Equations (3.23) and (3.24), roa_{ap} by Equation (3.25).

$$
\begin{aligned}
roa_{v_{tv1}} &= r_{v_{tv1}} / ac_{v_{tv1}} = 10.0/3.0 \approx 3.33 \\
roa_{v_{tv2}} &= r_{v_{tv2}} / ac_{v_{tv2}} = 8.6/3.0 \approx 2.87 \\
roa_{v_{tab1}} &= r_{v_{tab1}} / ac_{v_{tab1}} = 3.2/3.0 \approx 1.07 \\
roa_{v_{tab2}} &= r_{v_{tab2}} / ac_{v_{tab2}} = 3.2/3.0 \approx 1.07 \\
roa_{v_{tab3}} &= r_{v_{tab3}} / ac_{v_{tab3}} = 0.8/3.0 \approx 2.67 \\
roa_{t_{tv}} &= roa_{root_{tv}} = max(roa_{v_{tv1}}, roa_{v_{tv2}}) \\
&= max(3.33, 2.87) = 3.33 \\
roa_{t_{tab}} &= roa_{root_{tab}} = roa_{v_{tab1}} + roa_{v_{tab2}} + roa_{v_{tab3}} \\
&= 1.07 + 1.07 + 2.67 \approx 4.81 \\
roa_{ap} &= roa_{t_{tv}} + roa_{t_{tab}} = 3.33 + 4.81 \approx 8.14
\end{aligned}
$$

From the metric values in the node level, we can see that attacking the TV has higher success probability, lower cost and mean-time-to-compromise but lower

impact, risk and return-on-attacks than attacking the tablet because the attacker can exploit either vulnerability in TV to gain the root permission but needs to exploit all vulnerabilities in the tablet for the root permission. Thus the TV is easier to compromise. Besides, as the TV is also the entry point, we should protect the TV at first in order to prevent the attacker from breaking into the network or to make the attacker harder to break in.

We assume patching is used to fix the software bug existing in the TV. One mechanism is to patch v_{tv1} and denoted as $Defence_{v_{tv1}}$ while another is to patch v_{tv2} and denoted as $Defence_{v_{tv2}}$. We modify the vulnerability information for the TV, reconstruct the IoT network using the IoT Generator, compute the three-layered HARM and calculate the metric values after patching either v_{tv1} or v_{tv2}. The results calculated in the security analysis phase are shown in Table 3.5.

Table 3.5: Security analysis of the home network.

Metric / Mechanism	asp_{ap}	ac_{ap}	aim_{ap}	$mttc_{ap}$	r_{ap}	roa_{ap}
No defence	0.51	12.0	28.0	236.11	24.4	8.14
$Defence_{v_{tv1}}$	**0.44**	12.0	28.0	319.44	**23.0**	**7.67**
$Defence_{v_{tv2}}$	0.51	12.0	28.0	319.44	24.4	8.14

For both $Defence_{v_{tv1}}$ and $Defence_{v_{tv2}}$, $mttc_{ap}$ increases, which means both mechanisms are effective to extend the mean-time-to-compromise, while ac_{ap} and aim_{ap} do not change as the attack cost and impact values of v_{tv1} and v_{tv2} are the same respectively. For asp_{ap}, as exploiting v_{tv1} has higher success probability than exploiting v_{tv2}, deploying $Defence_{v_{tv1}}$ decreases the attack success probability more than deploying $Defence_{v_{tv2}}$. For r_{ap} and roa_{ap}, deploying $Defence_{v_{tv1}}$ causes lower risk and less gain than deploying $Defence_{v_{tv2}}$. If the defender is only able to deploy one mechanism to protect the TV (e.g., the manufacturer can only create one patch due to time and cost), $Defence_{v_{tv1}}$ should be chosen as it is more effective in reducing the attack success probability, risk and attacker's gain.

3.3.1.7 Summary

Using the framework, one can find potential attack paths, decide which devices included in the paths should be protected at first and compare the effectiveness of different device-level mechanisms based on the evaluation of various security metrics. As a result, one can choose the most effective device-level defence mechanisms for specific devices.

3.3.2 Node Controlling in Wearable Healthcare Monitoring

The emerging IoT has provided many benefits to the improvement of e-health applications. One application is the vital sign monitoring in hospitals [73], which uses wireless sensing technology to provide continuous monitoring for patients. As the data collected from the patients is sensitive, security threats may put a patient into a critical condition (e.g., lack of treatment or wrong treatment).

3.3.2.1 Scenario Description

We consider the wireless body area network (WBAN) which has been widely applied in wearable healthcare monitoring. It allows the vital physiological parameters of patients to be collected by wearable or implantable sensors and transmitted using short-range wireless communication techniques (e.g., IEEE 802.15.4 [3] or ZigBee [72]). In the WBAN, communications can be divided into two parts: intra-body and extra-body [79]. The intra-body communication network transmits data between the monitor sensors placed on the human body and the coordinator device (which is in charge of collecting data from monitor sensors and sending it to the external network). The extra-body communication network transmits data between the coordinator device and an external network (e.g., the hospital network providing local data processing and remote access via the Internet).

3.3.2.2 System Model

We use the intra-body communication in the WBAN in Figure 3.4 as an example. It shows 9 sensor nodes placed on the human body along with a coordinator device (e.g., PDA). The network is a heterogeneous network as nodes have different applications to measure different health data. For example, sensor node sn_1 measures the heart rate and the electrocardiogram (ECG) and sn_9 senses the blood oxygen.

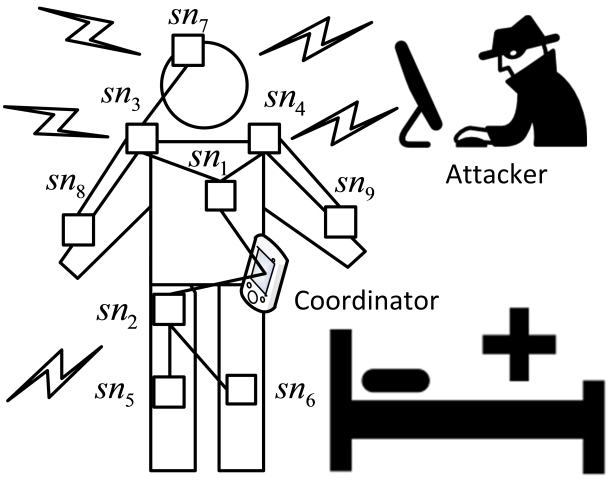


Figure 3.4: An intra-body communication network in the WBAN.

We assume a tree-based routing protocol is used for the intra-body communication [80] and the network topology does not change. Communications between sensor nodes and the coordinator device are single-hop or multi-hop. Data packets are sent to the coordinator device at pre-determined times or immediately when an emergency event occurs. Each sensor node runs the same operating system (e.g., TinyOS 2.x) with different applications and has a buffer overflow vulnerability in the operating system [52]. The coordinator device receives all information from the sensors and provides an interface towards the hospital network. A key management scheme is used to protect the data confidentiality, data integrity and data authentication [129].

3.3.2.3 Attacker Model

We assume the attacker's goal is to compromise a sensor node that stores critical data on it and manipulate the content of the data packets sent from the node. The attacker model describes the attacker's capabilities as follows.

1. The attacker is able to get into the hospital. However, as sensors are deployed on the human body, it is difficult for an attacker to physically access nodes without being detected. Thus the attacker can only communicate with the sensor nodes in its radio range.

2. The attacker has a laptop-class device with the radio module to send malicious packets. He can exploit the buffer overflow vulnerability targeting the operating system to compromise a sensor node within an accepted time. Once a node is compromised, the attacker has full control (e.g., steal cryptographic keys, obtain routing table, inject and run arbitrary code). He can also reprogram the compromised node into a malicious node and exploit it to compromise other nodes.

3. The coordinator device is well-protected and only allows the authenticated sensors to communicate with it such that the attacker cannot easily compromise the coordinator.

3.3.2.4 Data Processing

Based on our subnet classification method (i.e., the communication protocol), we introduce one subnet for the whole network as all sensor nodes use the radio communication. The subnet is denoted as s_{wban} which consists of the coordinator device denoted as t_{coord} and sensor nodes denoted as t_{sn_i} ($i \in \{1,2,...,9\}$). In the IoT Generator, the IoT network is represented as IoT_2 = (S_2, T_2, V_2) where S_2=$\{s_{wban}\}$, T_2=$\{t_{coord}, t_{sn_1}, ..., t_{sn_9}\}$ and V_2=$\{v_{sn}\}$. Here, v_{sn} represents the same buffer overflow vulnerability on each sensor node.

We choose the following security metrics to be used in the security analysis phase: attack success probability, attack cost, attack impact, mean-time-to-compromise, risk and return-on-attacks. We make assumptions about the metric values of vulnerability v_{sn}. This vulnerability allows an attacker to exploit within the communication range of the sensor node without any authentication and has low access complexity. However, it requires a radio module to communicate with the sensors. Thus we use high attack success probability and medium attack cost. After exploiting the vulnerability, the attacker has full control of the sensor node. Thus we use the maximum attack impact. We also assume the compromise rate as once per week as the attacker needs to be in the hospital to get access to the nodes. Estimated values of the security metrics for the vulnerability are shown in Table 3.6.

Table 3.6: Metric values for the vulnerability in the intra-body communication network.

Metric / Vulnerability	asp_v	ac_v	aim_v	cr_v
v_{sn}	0.8	5.0	10.0	0.006

3.3.2.5 Security Model Generation and Visualisation

We use $IoT_2 = (S_2, T_2, V_2)$ as the input into the Security Model Generator and compute the three-layered HARM. The model is represented as $GSM_2 = (U_2, M_2, L_2, C_{U_2,M_2}, C_{M_2,L_2})$. As an example of an attack graph and an attack tree, we show U_2=($\{k, s_{wban}\}$, $\{k \rightarrow s_{wban}\}$) and $at_{t_{sn_1}}$=($\{v_{sn}\}$, $\{root_{sn}\}$, $c(root_{sn})$=$\{v_{sn}\}$, $g(root_{sn})$=$\emptyset$, $root_{sn}$), respectively.

We assume the attacker's goal is to compromise sn_1 which measures the heart rate and the ECG information and manipulate the data sent from it to cause wrong treatment. The attacker is supposed to take either sn_4 or sn_9 as the access point by compromising it and exploiting it to compromise other nodes. Figure 3.5 shows the visualised attack paths in the network. As each node has the same vulnerability, we show only one v_{sn} in the lower layer. By exploiting

the vulnerability, the attacker is able to compromise a series of nodes and control them for malicious purpose.

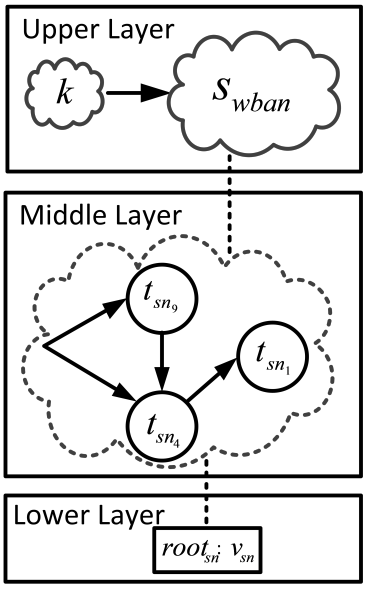


Figure 3.5: Attack paths in the intra-body communication network.

3.3.2.6 Security Analysis and Model Updates

We calculate the values of security metrics in the node, attack path and network levels. Network level metrics are denoted as ASP_2, AC_2, AIM_2 and $MTTC_2$. We define ap_1=(t_{sn_9}, t_{sn_4}, t_{sn_1}) and ap_2=(t_{sn_4}, t_{sn_1}).

Attack success probability: as each sensor has only one vulnerability, we calculate $asp_{t_{sn_i}}$ by Equation (3.2). We also calculate asp_{ap_1} and asp_{ap_2} by Equation (3.3). ASP_2 is calculated using Algorithm 1 in which the SHARPE output is 0.74.

$$asp_{t_{sn_i}} = asp_{root_{sn_i}} = 0.8$$

$$asp_{ap_1} = asp_{t_{sn_9}} * asp_{t_{sn_4}} * asp_{t_{sn_1}} = 0.8 * 0.8 * 0.8 = 0.512$$

$$asp_{ap_2} = asp_{t_{sn_4}} * asp_{t_{sn_1}} = 0.8 * 0.8 = 0.64$$

Attack cost: we calculate $ac_{t_{sn_i}}$ by Equation (3.6). We also calculate ac_{ap_1} and

ac_{ap_2} by Equation (3.7), AC_2 by Equation (3.8).

$$
\begin{aligned}
ac_{t_{sn_i}} &= ac_{root_{sn_i}} = 5.0 \\
ac_{ap_1} &= ac_{t_{sn_9}} + ac_{t_{sn_4}} + ac_{t_{sn_1}} = 5.0 + 5.0 + 5.0 = 15.0 \\
ac_{ap_2} &= ac_{t_{sn_4}} + ac_{t_{sn_1}} = 5.0 + 5.0 = 10.0 \\
AC_2 &= min(ac_{ap_1}, ac_{ap_2}) = min(15.0, 10.0) = 10.0
\end{aligned}
$$

Attack impact: we calculate $aim_{t_{sn_i}}$ by Equation (3.10). We also calculate aim_{ap_1} and aim_{ap_2} by Equation (3.11), AIM_2 by Equation (3.12).

$$
\begin{aligned}
aim_{t_{sn_i}} &= aim_{root_{sn_i}} = 10.0 \\
aim_{ap_1} &= aim_{t_{sn_9}} + aim_{t_{sn_4}} + aim_{t_{sn_1}} = 10.0 + 10.0 + 10.0 = 30.0 \\
aim_{ap_2} &= aim_{t_{sn_4}} + aim_{t_{sn_1}} = 10.0 + 10.0 = 20.0 \\
AIM_2 &= max(aim_{ap_1}, aim_{ap_2}) = max(30.0, 20.0) = 30.0
\end{aligned}
$$

Mean-time-to-compromise: we calculate $mttc_{ap_1}$and $mttc_{ap_2}$ by Equations (3.13) and (3.15), $MTTC_2$ by Equation (3.16).

$$
\begin{aligned}
mttc_{ap_1} &= mttc_{t_{sn_9}} + mttc_{t_{sn_4}} + mttc_{t_{sn_1}} = 1/cr_{t_{sn_9}} + 1/cr_{t_{sn_4}} + 1/cr_{t_{sn_1}} \\
&\approx 166.67 + 166.67 + 166.67 \approx 500.0 \\
mttc_{ap_2} &= mttc_{t_{sn_4}} + mttc_{t_{sn_1}} = 1/cr_{t_{sn_4}} + 1/cr_{t_{sn_1}} \\
&\approx 166.67 + 166.67 \approx 333.33 \\
MTTC_2 &= min(mttc_{ap_1}, mttc_{ap_2}) = min(500.0, 333.33) = 333.33
\end{aligned}
$$

Risk: we calculate $r_{v_{sn}}$ by Equation (3.17), $r_{t_{sn_i}}$ by Equation (3.19). We also

calculate r_{ap_1} and r_{ap_2} by Equation (3.20), R_2 by Equation (3.21).

$$
\begin{aligned}
r_{v_{sn}} &= asp_{v_{sn}} * aim_{v_{sn}} = 0.8 * 10.0 = 8.0 \\
r_{t_{sn_i}} &= r_{root_{sn_i}} = r_{v_{sn}} = 8.0 \\
r_{ap_1} &= r_{t_{sn_9}} + r_{t_{sn_4}} + r_{t_{sn_1}} = 8.0 + 8.0 + 8.0 = 24.0 \\
r_{ap_2} &= r_{t_{sn_4}} + r_{t_{sn_1}} = 8.0 + 8.0 = 16.0 \\
R_2 &= max(r_{ap_1}, r_{ap_2}) = max(24.0, 16.0) = 24.0
\end{aligned}
$$

Return-on-attacks: we calculate $roa_{v_{sn}}$ by Equation (3.22), $roa_{t_{sn_i}}$ by Equation (3.24). We also calculate roa_{ap_1} and roa_{ap_2} by Equation (3.25), ROA_2 by Equation (3.26).

$$
\begin{aligned}
roa_{v_{sn}} &= r_{v_{sn}} / ac_{v_{sn}} = 8.0/5.0 = 1.6 \\
roa_{t_{sn_i}} &= roa_{root_{sn_i}} = roa_{v_{sn}} = 1.6 \\
roa_{ap_1} &= roa_{t_{sn_9}} + roa_{t_{sn_4}} + roa_{t_{sn_1}} = 1.6 + 1.6 + 1.6 = 4.8 \\
roa_{ap_2} &= roa_{t_{sn_4}} + roa_{t_{sn_1}} = 1.6 + 1.6 = 3.2 \\
ROA_2 &= max(roa_{ap_1}, roa_{ap_2}) = max(4.8, 3.2) = 4.8
\end{aligned}
$$

From the metric values in the attack path level, we can see that exploiting ap_2 to reach the target has higher success probability, lower cost and mean-time-to-compromise but lower impact, risk and return-on-attacks than exploiting ap_1 as there are more nodes in ap_1 which need to be compromised by the attacker. Thus protecting nodes in ap_2 is more effective in blocking the attacker to reach the target.

In terms of the defence mechanism for the buffer overflow, we can deploy the method of address space layout randomisation (ASLR) for the node, denoted as $Defence_{ASLR}$. The ASLR method is based upon the low chance of an attacker guessing locations of randomly placed areas, thus enhancing the security by increasing the search space. We make the assumptions on the metric values of

the vulnerability after deploying the ASLR in Table 3.7. We decrease the attack success probability and compromise rate as the method can only complicate the attack but not eliminate the vulnerability. The method does not affect the attack cost and impact as the tools are the same and the impact measures the potential loss after the vulnerability is exploited.

Table 3.7: Metric values for the vulnerability in the intra-body communication network after the deployment of the defence mechanism.

Mechanism \ Metric	asp_v	ac_v	aim_v	cr_v
$Defence_{ASLR}$	0.5	5.0	10.0	0.003

We assume the defender wants to deploy the ASLR defence mechanism on one device in the attack path ap_2 because of the budget limit. We modify the vulnerability information for each sensor node, reconstruct the IoT network using the IoT Generator, compute the three-layered HARM and calculate the metric values. From the metric values in the network level shown in Table 3.8, we can assess the effectiveness of the mechanism deployed on each node.

Table 3.8: Security analysis of the intra-body communication network.

Mechanism \ Metric	ASP_2	AC_2	AIM_2	$MTTC_2$	R_2	ROA_2
No defence	0.74	10.0	30.0	333.33	24.0	4.8
$Defence_{ASLR}$ on sn_4	0.56	10.0	30.0	499.99	21.0	4.2
$Defence_{ASLR}$ on sn_1	**0.46**	10.0	30.0	499.99	21.0	4.2

For $Defence_{ASLR}$ on both sn_4 and sn_1, ASP_2, R_2 and ROA_2 decrease while $MTTC_2$ increases. ASP_2 is lower when using the ASLR on sn_1 since sn_4 and sn_1 have different locations in the network. AC_2 and AIM_2 do not change as the metric values do not change before and after the defence mechanism. Thus, based on the analysis results, we can see that protecting sn_1 is more effective against the attack.

3.3.2.7 Summary

Using the framework, one can compare the severity of multiple potential attack paths and the effectiveness of specific device-level mechanisms deployed for different devices. This helps to decide which devices should be protected at first.

3.3.3 Traffic Analysis in Environment Monitoring

Among the IoT application domains, the habitat and environment monitoring has received a growing interest as it is essential for studying and making efficient use of our environment. As the first step of the analysis, sensor networks are used to collect data from the environment. As sensor networks are usually deployed in an open field with little human interaction, they are prone to failures due to extreme climatic conditions or various malicious attacks.

3.3.3.1 Scenario Description

Wireless Sensor Networks (WSNs) have been widely used in the IoT environment monitoring applications as the WSNs are well-suited for long-term environmental sensing for the IoT applications. With the WSNs, environmental monitoring includes both indoor and outdoor applications [81]. One outdoor application is the habitat monitoring which requires a large number of low-cost sensor nodes and a gateway node (i.e., the sink) deployed in a given landscape. Sensor nodes are responsible for data acquisition while the gateway node connects to the remote servers via the Internet.

3.3.3.2 System Model

We consider a WSN with 1000 sensor nodes and one sink deployed in an open and unattended field shown in Figure 3.6. The network is a homogeneous network as each sensor node has the same application for sensing the temperature and humidity of the environment.

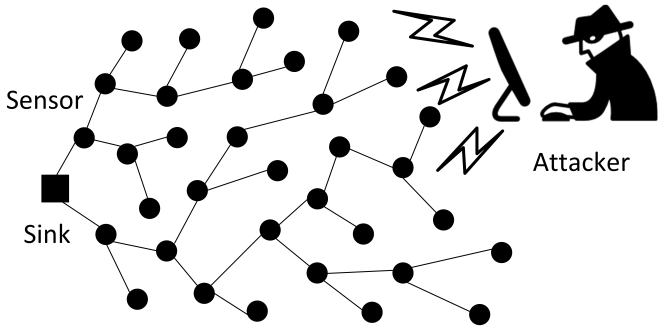


Figure 3.6: A wireless sensor network.

We assume sensor nodes and the sink are static after deployment. Sensor nodes self-organise and form a routing tree which is rooted at the sink [140]. Each sensor has a transmission range of r meters and uses bidirectional wireless communication. Communications between the sink and sensor nodes are single-hop or multi-hop. Sensor nodes periodically send packets to the sink (e.g., every 10 minutes). Data packets are encrypted by employing a pair-wise key scheme [44]. The sink is connected to the Internet and becomes the gateway between the sensor network and the Internet.

3.3.3.3 Attacker Model

We assume the attacker's goal is to destroy the sink physically after finding its location. As the sink is the central point of failure, destroying it will make the whole network unavailable for sending data to the remote servers. The attacker model is based on [41] which describes the attacker's capabilities as follows.

1. In the wireless communication, radio links are insecure. We assume an attacker can eavesdrop on radio transmissions by distributing a wireless monitoring device in the area of interest. The transmission range of the monitoring device is larger than the transmission range of a sensor node (e.g., $3r$ meters) but does not cover the entire network.

2. The attacker can physically move from one location to another location in the network but cannot monitor the entire network.

3. Each node routes packets along a fixed path to the sink using wireless communication. Thus the attacker can launch a rate-monitoring attack to deduce the location of the sink by monitoring the packet sending rate of nodes and moving towards the nodes with higher rates.

4. As the sink is in an open environment, the attacker can physically damage it once he discovers its location.

3.3.3.4 Data Processing

As all sensor nodes in the network are identical, we introduce one subnet for the whole network, denoted as s_{wsn}. According to the attacker's capabilities in Section 3.3.3.3, we denote the vulnerability of the sensor node described in capability 3 as v_{sn} and the vulnerability of the sink described in capability 4 as v_{sink}. In the IoT Generator, the IoT network is represented as $IoT_3 = (S_3, T_3, V_3)$ where S_3=$\{s_{wsn}\}$, T_2=$\{t_{sink}, t_{sn_1}, ..., t_{sn_{1000}}\}$ and V_3=$\{v_{sn}, v_{sink}\}$.

We choose the following security metrics to be used in the security analysis phase: attack success probability, attack cost, attack impact, mean-time-to-compromise, risk, return-on-attacks and number of attack paths. We make assumptions about the metric values of the vulnerabilities. For v_{sn}, sensor nodes are deployed in an open field, thus allowing easy access to them. However, an attacker needs to purchase and distribute a special device to monitor the packet sending rate. Thus we use high attack success probability and high attack cost. By using the vulnerability, the attacker may be able to discover the position of the base station. We assume a medium attack impact. For v_{sink}, as the sink is deployed in an open field, the attacker can easily damage it physically once he knows the location. So we use high attack success probability and medium attack cost respectively. Once the sink is damaged, the data gathered from sensor nodes cannot be delivered to remote servers. Thus we use the maximum attack impact. Besides, for both v_{sn} and v_{sink}, we use a compromise rate of once per week as the sensor network is deployed in remote areas and it is not easy for an attacker to

get to the areas. The estimated values for security metrics of the vulnerabilities are shown in Table 3.9.

Table 3.9: Metric values for each vulnerability in the WSN.

Metric / Vulnerability	asp_v	ac_v	aim_v	cr_v
v_{sn}	0.9	8.0	4.0	0.006
v_{sink}	0.9	5.0	10.0	0.006

3.3.3.5 Security Model Generation and Visualisation

We use $IoT_3 = (S_3, T_3, V_3)$ as the input into the Security Model Generator and compute the three-layered HARM. The model is represented as $GSM_3 = (U_3, M_3, L_3, C_{U_3,M_3}, C_{M_3,L_3})$. As an example of an attack graph and an attack tree, we show U_3=($\{k, s_{wsn}\}$, $\{k \rightarrow s_{wsn}\}$) and $at_{t_{sn_1}}$=($\{v_{sn}\}$, $\{root_{sn}\}$, $c(root_{sn})$=$\{v_{sn}\}$, $g(root_{sn})$=$\emptyset$, $root_{sn}$), respectively.

The attacker is assumed to access one sensor node (e.g., sn_{999} deployed at the edge of the network). Figure 3.7 shows the visualised attack path in the network. As each sensor has the same vulnerability, we show only one v_{sn} in the lower layer. By exploiting the vulnerabilities, the attacker is able to move along the nodes with a higher packet sending rate and discover the location of the sink.

3.3.3.6 Security Analysis and Model Updates

As each node (i.e., a sensor node or the sink) has only one vulnerability which can be exploited by the attacker, metric values in the vulnerability level equal to the values in the node level. There is only one attack path captured in the HARM. Thus $NAP = 1$. We list the nodes in the attack path as ap=($t_{sn_{999}}, t_{sn_{499}}, t_{sn_{249}}, t_{sn_{124}}, t_{sn_{61}}, t_{sn_{30}}, t_{sn_{14}}, t_{sn_6}, t_{sn_2}, t_{sink}$).

Attack success probability: we calculate ASP_3 using Algorithm 1 in which the SHARPE output is approximately 0.35.

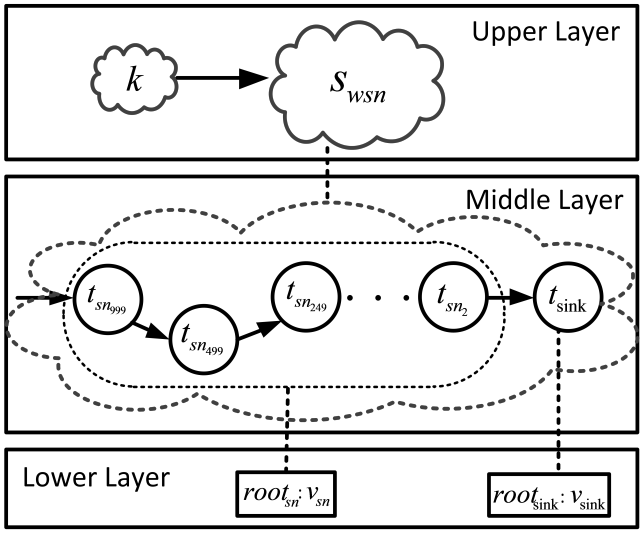


Figure 3.7: Attack path in the WSN.

Attack cost: we calculate AC_3 by Equations (3.6), (3.7) and (3.8).

$$\begin{aligned} AC_3 &= ac_{ap} = ac_{t_{sn_{999}}} + ac_{t_{sn_{499}}} + ... + ac_{t_{sink}} \\ &= ac_{root_{sn_{999}}} + ac_{root_{sn_{499}}} + ... + ac_{root_{sink}} = 77.0 \end{aligned}$$

Attack impact: we calculate AIM_3 by Equations (3.10), (3.11) and (3.12).

$$\begin{aligned} AIM_3 &= aim_{ap} = aim_{t_{sn_{999}}} + aim_{t_{sn_{499}}} + ... + aim_{t_{sink}} \\ &= aim_{root_{sn_{999}}} + aim_{root_{sn_{499}}} + ... + aim_{root_{sink}} = 46.0 \end{aligned}$$

Mean-time-to-compromise: we calculate $MTTC_3$ by Equations (3.13), (3.15) and (3.16).

$$\begin{aligned} MTTC_3 &= mttc_{ap} = mttc_{t_{sn_{999}}} + mttc_{t_{sn_{499}}} + ... + mttc_{t_{sink}} \\ &= 1/cr_{t_{sn_{999}}} + 1/cr_{t_{sn_{499}}} + ... + 1/cr_{t_{sink}} \approx 1666.66 \end{aligned}$$

Risk: we calculate $r_{v_{sn}}$ and $r_{v_{sink}}$ by Equation (3.17), R_3 by Equations (3.19),

(3.20) and (3.21).

$$
\begin{aligned}
r_{v_{sn}} &= asp_{v_{sn}} * aim_{v_{sn}} = 0.9 * 4.0 = 3.6 \\
r_{v_{sink}} &= asp_{v_{sink}} * aim_{v_{sink}} = 0.9 * 10.0 = 9.0 \\
R_3 = r_{ap} &= r_{t_{sn_{999}}} + r_{t_{sn_{499}}} + ... + r_{t_{sink}} \\
&= r_{root_{sn_{999}}} + r_{root_{sn_{499}}} + ... + r_{root_{sink}} = 41.4
\end{aligned}
$$

Return-on-attacks: we calculate $roa_{v_{sn}}$ and $r_{v_{sink}}$ by Equation (3.22), ROA_2 by Equations (3.24), (3.25) and (3.26).

$$
\begin{aligned}
roa_{v_{sn}} &= r_{v_{sn}} / ac_{v_{sn}} = 3.6/8.0 = 0.45 \\
roa_{v_{sink}} &= r_{v_{sink}} / ac_{v_{sink}} = 9.0/5.0 = 1.8 \\
ROA_3 = roa_{ap} &= roa_{t_{sn_{999}}} + roa_{t_{sn_{499}}} + ... + roa_{t_{sink}} \\
&= roa_{root_{sn_{999}}} + roa_{root_{sn_{499}}} + ... + roa_{root_{sink}} = 5.85
\end{aligned}
$$

In terms of the defence mechanism, we can deploy the multi-parent routing (MPR) scheme for the sensor node proposed in [41], denoted as $Defence_{MPR}$. When forwarding a packet, the node randomly selects one of its parent nodes to forward the packet. Thus the attacker needs more time to guess which path to follow in order to reach the sink. We make the assumptions on the metric values of the vulnerability v_{sn} after deploying the MPR scheme in Table 3.10. We decrease the attack success probability and compromise rate as the method complicates the attack but does not eliminate the vulnerability. The method does not affect the attack cost impact as the tools are the same and the impact measures the loss after the vulnerability is exploited.

Table 3.10: Metric values for the vulnerability v_{sn} in the WSN after defence.

Metric / Mechanism	asp_v	ac_v	aim_v	cr_v
$Defence_{MPR}$	0.6	8.0	4.0	0.003

We use the framework to analyse whether the defence mechanism is effective or not based on the network-level security metrics. Figure 3.8 shows the new attack paths in the network. After deploying the MPR scheme, the three-layered HARM captures multiple attack paths from a sensor node (i.e., the break-in point) to the sink. We compare the metrics values before and after the deployment of the defence mechanism in Table 3.11.

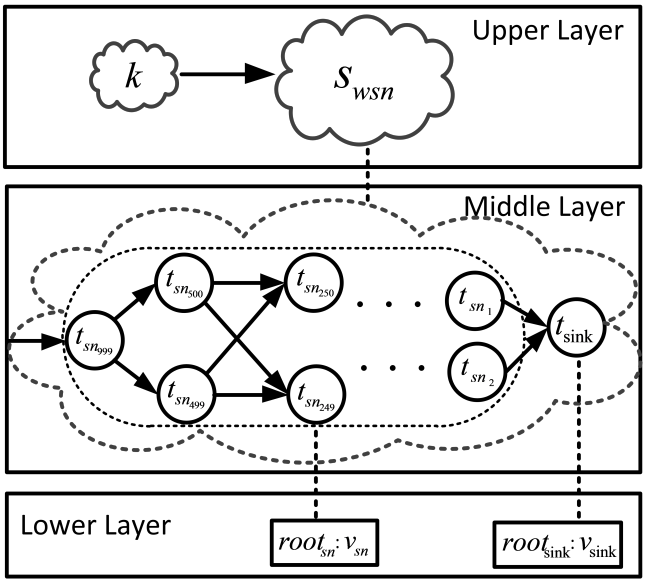


Figure 3.8: Attack paths in the WSN after the deployment of the defence mechanism.

Table 3.11: Security analysis of the WSN.

Metric / Mechanism	ASP_3	AC_3	AIM_3	$MTTC_3$	R_2	ROA_2	NAP
No defence	0.35	77.0	46.0	1666.66	41.4	5.85	1
$Defence_{MPR}$	**0.27**	77.0	46.0	**3166.66**	**30.6**	**4.5**	**256**

After deploying the MPR scheme, ASP_3, R_3 and ROA_3 decrease while $MTTC_3$ increases, which indicates the scheme is effective to lower the attack success probability, risk, attacker's gain and extend the mean-time-to-compromise. AC_3 and AIM_3 do not change as the attack cost and impact values do not change before and after the defence mechanism. Besides, NAP increases significantly to confuse

the attacker. Thus, we can conclude the network-level defence mechanism is effective against the traffic analysis attack.

3.3.3.7 Summary

Using the framework, one can assess the effectiveness of network-level defence mechanisms deployed for the network based on the security metrics.

Chapter 4

Security Analysis of the Software-Defined IoT

The SDN is an emerging technology that defines new ways to manage networks [63, 76, 105, 126]. In the SDN-based architecture, the control logic is decoupled from the switches and routers and implemented in a logically centralised controller; the controller communicates with the data forwarding devices via the southbound application programming interface (API) and also provides the programmability of network applications via the northbound API. OpenFlow is the most widely used southbound API which provides the specifications for the implementation of OpenFlow switches (including the OpenFlow ports, tables, channel and protocol) [4]. Besides, the SDN is also foreseen as a promising paradigm for managing large-scale and complex IoT networks.

In this chapter, we propose a novel approach to address the security issues arising from the non-patchable vulnerabilities in the IoT devices. We change the attack surface of the IoT network with the support of SDN functions to increase the attack efforts by the attackers. Here, we focus on exploitable vulnerabilities as our attack surface. We discuss related work on the existing SDN solutions for the IoT and security issues of the SD-IoT, present our proposed defence

mechanisms and reconfiguration algorithms for changing the topology of the IoT based on the SDN functions and perform simulations to validate the efficacy of the mechanisms.

To the best of our knowledge, this work is the first approach to model and assess the security of the SD-IoT via the graphical security model while taking into account the performance of the network. The main contributions are summarised as follows:

- Design two proactive defence mechanisms that reconfigure the topology of the IoT network based on the SDN functions (Section 4.1);
- Develop optimal and heuristic reconfiguration algorithms for the mechanisms (Section 4.1.4);
- Conduct security and performance analysis of the SD-IoT network using the graphical security model and various metrics (Sections 4.2.1 and 4.2.2);
- Carry out experimental analysis via simulations to investigate the effectiveness of our proposed defence mechanisms (Sections 4.2.4 and 4.2.5).

4.1 Proactive Defence Mechanisms

In the IoT, heterogeneous devices can have different hardware and software vendors providing a variety of services. Some software vendors provide updates periodically; some vendors are unable to do so according to time and cost for creating patches. Besides, legacy devices in their end-of-life phases cannot receive updates from their vendors. Hence, non-patchable vulnerabilities leave the devices exploitable by attackers. To deal with this problem, such as forever-day vulnerabilities, we change the attack surface of the IoT by reconfiguring the network topology. We consider the following two cases as some software vendors may not provide patches for their products, leaving some devices not patchable.

- **Case I:** an IoT network with a mix of (i) patchable nodes (which are nodes with patchable vulnerabilities) and (ii) non-patchable nodes (which are nodes with non-patchable vulnerabilities).

- **Case II:** an IoT network with only non-patchable nodes in which (i) some nodes with "hard-to-exploit" vulnerabilities (e.g., low attack success probability) (ii) some nodes with "easy-to-exploit" vulnerabilities (e.g., high attack success probability).

In the real world, there are more complex cases. For example, the IoT devices can have a mix of patchable and non-patchable vulnerabilities; non-patchable IoT devices can have a variety of vulnerabilities in which some are easy to exploit while others are harder to exploit. However, this work aims to demonstrate the use of the graphical security model to analyse the effectiveness of reconfiguration to achieve better security. More cases can be investigated in the future work.

4.1.1 Scenario Description

Wireless Sensor Networks (WSNs) have been widely used in the IoT environment monitoring applications because the WSNs are well-suited for long-term environmental sensing for the IoT applications [78, 103, 123]. Besides, as sensor networks are usually deployed in an open field with little human interaction, they are prone to various malicious attacks. The smart home and wearable health monitoring scenarios in Chapter 3 are not used in this approach because many SDN solutions for the IoT only support the WSNs for the flexible network management. Thus, we use the WSNs for both Case I and Case II.

4.1.2 System Model

In the system model, we consider a software-defined sensor network for smart environment sensing. We make the following assumptions on the network components. The network contains a base station and two types of heterogeneous

outdoor sensor nodes with different applications for sensing the environment (e.g., noise, weather, air quality). The senor nodes collect data and periodically deliver them to the base station via single hop or multiple hops. The base station sends the data to the Cloud or remote servers for processing via the Internet.

We make the following assumptions on the network deployment. The network is deployed in a specific area of our interest (e.g., a campus). Based on [123], sensor nodes can be placed on the street lights, traffic lights, bus stops or outside of the buildings to support particular services. Street lights and traffic lights can provide power supply for the attached sensor nodes. The base station is placed on the ground without being noticeable. One sensor node at the edge of the network communicates with the base station directly or via a range extender.

We make the following assumptions on the specifications of the sensor nodes and the base station. Each sensor node is based on the microcontroller (e.g., CC2530) and configured with the ZigBee radio module [72] (based on IEEE 802.15.4 standard). The base station is implemented on the single-board computer attached with the 2.4 GHz RF transceiver (e.g., CC2500). We assume the SDN-WISE protocol designed in [46] is applied in the sensor network as the southbound API between sensor nodes and Controllers to achieve programmability. The Controller can be implemented using any programming language and placed in a remote server. A network virtualisation layer, named WISE-Visor, runs as the proxy between sensor nodes and the Controller. The base station acts as the gateway between sensor nodes and the Controller and implements an Adaptation layer to format control packets. After deployment, each sensor node runs a special protocol to find the next hop towards the Controller and stores this information in the flow table entry. Once the sensor node is able to communicate with the Controller, the flow table is continuously updated based on the commands sent by the Controller. We use a tree topology as the initial network topology because the tree topology has been widely used in the wireless sensor networks for data collection applications and supported by many communication protocols (e.g., ZigBee [72]).

Besides the above general assumptions, we make the assumptions on the specific system models for Case I and Case II in the following, respectively. Any types of the sensor nodes can be used in the system model. The following assumptions aim to give a more realistic example of how the approach can be applied to the real-world networks.

- **Case I:** a software-defined sensor network consists of two types of nodes, (i) noise sensor nodes and (ii) weather sensor and air quality sensor nodes; noise sensor nodes are patchable nodes while weather and air quality sensor nodes are non-patchable nodes.
- **Case II:** a software-defined sensor network consists of two types of non-patchable nodes, (i) noise sensor nodes and (ii) weather sensor nodes; noise sensor nodes have the "hard-to-exploit" vulnerability while weather sensor nodes have the "easy-to-exploit" vulnerability.

We make the following assumptions on the vulnerability information for both cases. In Case I and Case II, we assume sensor nodes only have known vulnerabilities. Unknown vulnerabilities (e.g., zero-day vulnerabilities) can be considered in the future work. We use two types of vulnerabilities: buffer overflow and no verification of updates. Buffer overflow attack is one of the most frequent attacks [38] and has been found in many IoT devices due to the coding flaws. Lack of update verification is also one of the most common design flaws in the IoT devices because of the limited resources. Both vulnerabilities allow remote attackers to execute arbitrary code after successful exploit and to launch further attacks (e.g., Distributed DoS attacks [147]).

In Case I, we assume the noise sensor node has two vulnerabilities in its noise sensing application: one is a buffer overflow vulnerability; another is that the application does not verify the integrity of the downloaded updates, which allows the attacker to execute arbitrary code via Trojan horse update [2]. We assume the weather and air quality sensor node has two buffer overflow vulnerabilities:

one is in the weather application and another is in the air quality application. In Case II, we assume both noise sensor nodes and weather sensor nodes have one buffer overflow vulnerability in their applications while the vulnerability in the noise sensing application is harder to exploit by the attacker. We summarise the types of nodes and the node vulnerability information in Table 4.1.

Table 4.1: Vulnerability information of sensor nodes in Case I and Case II.

Node	Application	Vulnerability Information	
Case I			
Noise sensor node	Noise	Buffer overflow	Patchable
		No verification of updates	Patchable
Weather and air quality sensor node	Weather	Buffer overflow	Non-patchable
	Air quality	Buffer overflow	Non-patchable
Case II			
Noise sensor node	Noise	Buffer overflow	Hard to exploit
Weather sensor node	Weather	Buffer overflow	Easy to exploit

4.1.3 Attacker Model

As the sensor network is deployed in the open area to monitor the environment, an attacker can easily get access to the sensor nodes and compromise them. We assume an outsider attacker in the attacker model. The outsider attacker is an unauthorised user who does not have permission to control the sensor network. The goal of the attacker is to compromise the base station. The assumptions of the attacker's capabilities are listed as follows. We use the attacker model for both Case I and Case II.

- The attacker is able to get access to a certain part of the area with the deployed sensor nodes and use one node as the entry point.
- The attacker has a laptop-class device configured with the IEEE 802.15.4 radio module. He can exploit any vulnerabilities in the applications of sensor nodes listed in Table 4.1. Once the node is compromised, the attacker can inject and run arbitrary codes. He can reprogram the node into

a malicious one, use it to compromise other nodes by sending malicious packets, and eventually compromise the base station through using the sensor nodes as his stepping-stones.

- It is hard for the attacker to compromise the base station directly because the attacker does not know the position of the base station and the base station only allows the authenticated sensors to communicate with it.

4.1.4 Reconfiguration Algorithms

For Case I and Case II, we propose two proactive defence mechanisms that reconfigure the IoT network topology based on SDN functions. We describe our mechanisms in the following.

- **Case I:** as the network contains a mix of patchable and non-patchable nodes, we reconfigure the topology of the initial network by maximising the number of patchable nodes along the route to the base station. As the vulnerabilities of the patchable nodes can be patched when the vendor releases the updates, the patchable nodes will have less number of vulnerabilities after patching. According to the system model, we maximise the number of noise sensor nodes with patchable vulnerabilities along the route to the base station.

- **Case II:** as the network contains only non-patchable nodes, we reconfigure the topology of the initial network by maximising the number of sensor nodes with vulnerabilities which are harder to exploit along the route to the base station. According to the system model, we maximise the number of noise sensor nodes with the "hard-to-exploit" vulnerability along the route to the base station.

We develop reconfiguration algorithms for the two proactive defence mechanisms, an optimal algorithm in Section 4.1.4.2 and two heuristic algorithms

in Section 4.1.4.3. The algorithms will be executed in the Controller. The current algorithms only apply to the nodes with one parent node. This restriction can be released in the future work to adapt the algorithms to other topologies. Besides, the algorithms apply to the following two system models: the network consisting of nodes only with patchable vulnerabilities and nodes only with non-patchable vulnerabilities or the network consisting of nodes only with "hard-to-exploit" vulnerabilities and nodes only with "easy-to-exploit" vulnerabilities.

4.1.4.1 General Notations

We list the notations used in the algorithms in the following.

- t_{bs}: the base station in the network
- t_{s_i}: the sensor node in the network ($i \in \{1,...,n\}$ where n is the number of sensor nodes)
- NT: a set of nodes in the network ($NT = \{t_{bs}, t_{s_1}, ..., t_{s_n}\}$)
- SE: a set of connections between nodes in NT ($SE \subseteq NT \times NT$)
- $se_{t_{s_i},t_{s_j}} \in SE$: the bidirectional connection between t_{s_i} and t_{s_j}
- NG: the graph of the network before reconfiguration ($NG = (NT, SE)$)
- $NG*$: the graph of the network after reconfiguration ($NG* = (NT*, SE*)$)
- H_{max}: the maximum hop count in NG
- h: a hop count in NG
- l: the reconfiguration limit which is the number of increased hops
- $CR_{t_{s_i}}$: the communication range of t_{s_i}
- $D_{t_{s_i},t_{s_j}}$: the distance between t_{s_i} and t_{s_j}
- $H_{t_{s_i}}$: the shortest hop count from t_{s_i} to t_{bs}

- $G_{t_{s_i}}$: a set of nodes that have the potential new connections with t_{s_i}
- $t_{p_{s_i}} \in t_{s_{iadj}}$: the current parent node of t_{s_i}
- $t_{tp_{s_i}}$: a node that temporarily stores the current parent node of t_{s_i}
- $t_{s_{ivuls}}$: a set of vulnerabilities of t_{s_i}
- $x_{t_{s_i}}$: 1 indicating t_{s_i} is patchable or has "hard-to-exploit" vulnerabilities or 0 indicating t_{s_i} is non-patchable or has "easy-to-exploit" vulnerabilities
- $X_{t_{s_i}}$: the maximum number of patchable nodes or nodes with "hard-to-exploit" vulnerabilities along the route from t_{s_i} to t_{bs}; initial value equals to $x_{t_{s_i}}$
- $Z_{t_{s_i}}$: a set of values storing the number of patchable nodes or the number of nodes with "hard-to-exploit" vulnerabilities along the route from t_{s_i} to t_{bs}
- $W_{t_{s_i}}$: a set of potential parent nodes of t_{s_i} giving the maximum number of patchable nodes or the maximum number of nodes with "hard-to-exploit" vulnerabilities along the route from t_{s_i} to t_{bs}
- $R(t_{s_i}, t_{bs})$: the satisfactory condition when there is a route from t_{s_i} to t_{bs}, returning *True*; otherwise returning *False*
- $L(t_{s_i}, t_{bs}, l)$: the satisfactory condition when the number of increased hops from t_{s_i} to t_{bs} via the new parent and from any node to t_{bs} via t_{s_i} is smaller than or equal to l, returning *True*; if any result is larger than l, returning *False*
- $F(t_{s_i})$: a function that changes the hop count of all child nodes of t_{s_i}

4.1.4.2 Optimal Method

We present the optimal reconfiguration algorithm in Algorithm 3. The algorithm gives the maximum number of patchable nodes in Case I or the

maximum number of nodes with "hard-to-exploit" vulnerabilities in Case II without changing the hop count of nodes in the network.

Algorithm 3: Calculation of the optimal topology

```
   Data: NG
   Result: NG*
1  begin
2      NG* ← NG
3      for each h∈{2,...,H_max} do
4          for each t_{s_i}∈{t_{s_1},...,t_{s_n}} ⊆ NT* do
5              if H_{t_{s_i}} == h then
6                  for each t_{s_j}∈{t_{s_1},...,t_{s_n}} do
7                      if t_{s_j} != t_{s_i} and H_{t_{s_j}} == h−1 and D_{t_{s_i},t_{s_j}} <= CR_{t_{s_i}} then
8                          Add t_{s_j} into G_{t_{s_i}}
9                      end
10                 end
11                 Sort G_{t_{s_i}} based on D_{t_{s_i},t_{s_k}} where t_{s_k} ∈ G_{t_{s_i}}
12                 for each t_{s_k}∈G_{t_{s_i}} do
13                     Add x_{t_{s_i}} + X_{t_{s_k}} into Z_{t_{s_i}}
14                 end
15                 X_{t_{s_i}} ← MAX(Z_{t_{s_i}})
16                 for each t_{s_u}∈G_{t_{s_i}} do
17                     if x_{t_{s_i}} + X_{t_{s_u}} == X_{t_{s_i}} then
18                         Add t_{s_u} into W_{t_{s_i}}
19                     end
20                 end
21                 if t_{p_{s_i}} ∉ W_{t_{s_i}} then
22                     for each t_{s_q}∈W_{t_{s_i}} do
23                         Remove es_{t_{p_{s_i}},t_{s_i}}
24                         t_{p_{s_i}} ← t_{s_q}
25                         Create es_{t_{s_q},t_{s_i}}
26                         break
27                     end
28                 end
29             end
30         end
31     end
32 end
```

In Algorithm 3, at first, we initialise the network after reconfiguration as the initial network (line 2). For each hop count starting from 2 to the maximum value (line 3), we go through each node t_{s_i} in the network (line 4). When the node has the current hop count (line 5), we check any other node t_{s_j} in the network whether

they have one less hop count and whether they are in the communication range of t_{s_i} (lines 6, 7). If all requirements are satisfied, we add t_{s_j} into a set $G_{t_{s_i}}$ (line 8). We sort $G_{t_{s_i}}$ based on the node's distance to t_{s_i} (line 11). For each node t_{s_k} in $G_{t_{s_i}}$, we calculate the sum of $x_{t_{s_i}}$ and $X_{t_{s_k}}$ and add the sum into a set $Z_{t_{s_i}}$ (lines 12, 13). We assign the maximum sum in $Z_{t_{s_i}}$ to $X_{t_{s_i}}$ (line 15). For each node t_{s_u} in $G_{t_{s_i}}$, we compare the sum of $x_{t_{s_i}}$ and $X_{t_{s_u}}$ with $X_{t_{s_i}}$ and add t_{s_u} into a set $W_{t_{s_i}}$ if the sum equals to $X_{t_{s_i}}$ (lines 16 - 18). We check whether the parent of t_{s_i} is in $W_{t_{s_i}}$ (line 21) and keep the current parent of t_{s_i} if it is already in $W_{t_{s_i}}$. Otherwise, we remove the link between t_{s_i} and its parent node, set t_{s_q} (the first node in $W_{t_{s_i}}$ with the closest distance to t_{s_i}) as the new parent node, create a link between t_{s_i} and t_{s_q} and break the inner loop of $W_{t_{s_i}}$ (lines 22 - 26).

4.1.4.3 Heuristic Method

In the optimal method, the hop count of each node in the network cannot be changed. In order to release the restriction, we introduce the reconfiguration limitation which is the number of increased hops after reconfiguration and develop two heuristic reconfiguration algorithms with the local optimal solution. The algorithms guarantee the new connection of a node to be a specific type of parent node in its communication range (i.e., a patchable parent node in Case I or a parent node with "hard-to-exploit" vulnerabilities) with the maximum hop count that satisfies the reconfiguration limitation towards the base station. However, the maximum number of patchable nodes or nodes with "hard-to-exploit" vulnerabilities along the path to the base station is not guaranteed. Besides, the algorithms do not maximise the length of the path from the node to the base station under the reconfiguration limitation. The algorithms that take consideration of maximising the number of patchable nodes or nodes with "hard-to-exploit" vulnerabilities and the length of the path under the reconfiguration limitation can be developed in the future work. Besides, when applying the heuristic algorithms on the networks repeatedly, changes in the reconfigured network will be identified when taking it as the input of the algorithms again.

Both algorithms converge when there are no available parent nodes that can be connected to (i.e., the patchable nodes or nodes with "hard-to-exploit" vulnerabilities). We present the heuristic reconfiguration algorithm for Case I in Algorithm 4.

Algorithm 4: Calculation of the reconfigured topology for Case I

```
   Data: NG
   Result: NG*
1  begin
2      NG* ← NG
3      for each t_{s_i} ∈ {t_{s_1}, ..., t_{s_n}} ⊆ NT* do
4          if t_{p_{s_i}} is non-patchable then
5              for each t_{s_j} ∈ {t_{s_1}, ..., t_{s_n}} do
6                  if t_{s_j} != t_{s_i} and t_{s_j} is patchable and D_{t_{s_i},t_{s_j}} <= CR_{t_{s_i}} then
7                      Add t_{s_j} into G_{t_{s_i}}
8                  end
9              end
10             Sort G_{t_{s_i}} based on H_{t_{s_k}} reversely where t_{s_k} ∈ G_{t_{s_i}}
11             for each t_{s_k} ∈ G_{t_{s_i}} do
12                 Remove es_{t_{ps_i},t_{s_i}}
13                 t_{tp_{s_i}} ← t_{p_{s_i}}
14                 t_{p_{s_i}} ← t_{s_k}
15                 Create es_{t_{s_k},t_{s_i}}
16                 if R(t_{s_i}, t_{bs}) is True and L(t_{s_i}, t_{bs}, l) is True then
17                     H_{t_{s_i}} ← H_{t_{s_k}} + 1
18                     F(t_{s_i})
19                     break
20                 else
21                     Remove es_{t_{s_k},t_{s_i}}
22                     t_{p_{s_i}} ← t_{tp_{s_i}}
23                     Recover es_{t_{ps_i},t_{s_i}}
24                 end
25             end
26         end
27     end
28 end
```

In Algorithm 4, at first, we initialise the network after reconfiguration as the initial network (line 2). For each node t_{s_i} in the network (line 3), we check whether its parent node is patchable or non-patchable (line 4). If the parent node is non-patchable, we check any other node t_{s_j} in the network whether they are

patchable and whether they are in the communication range of t_{s_i} (lines 5, 6). If all requirements are satisfied, we add t_{s_j} into a set $G_{t_{s_i}}$ (line 7). We sort $G_{t_{s_i}}$ based on the node's hop count to t_{bs} reversely (line 10). For each node t_{s_k} in $G_{t_{s_i}}$, we remove the link between t_{s_i} and its parent node, store the current parent in $t_{tp_{s_i}}$, set t_{s_k} as the new parent node and create a link between t_{s_i} and t_{s_k} (lines 12 - 15). If there is a route from t_{s_i} to t_{bs} via t_{s_k} and all paths from child nodes of t_{s_i} to t_{bs} do not increase certain number of hops to t_{bs} (line 16), we update the hop count of t_{s_i} and all its child nodes and break the inner loop of $G_{t_{s_i}}$ (lines 17 - 19). Otherwise, we remove the link between t_{s_i} and its new parent node, recover the original parent node and its connection (lines 21 - 23) and go through next node in $G_{t_{s_i}}$.

We present the heuristic reconfiguration algorithm for Case II in Algorithm 5. This algorithm is similar as Algorithm 4. The differences are in lines 4 and 6: we check whether the parent node has "easy-to-exploit" vulnerabilities in line 4; we check whether the vulnerabilities of the node are "hard-to-exploit".

4.2 Simulations

We carry out simulations to validate the proposed mechanisms. We present simulation settings, simulation steps, security modelling and analysis of an example network and simulation results using the optimal and heuristic methods in this section.

4.2.1 Simulation Settings

In the simulations, we use a software-defined sensor network consisting of 100 sensor nodes and 1 base station. One sensor node at the edge of the network is connected to the base station directly or via the range extender. Sensor nodes are randomly deployed in a specific area (e.g., campus [103]). The area has a high density of outdoor sensor nodes. In specific, we define a simulation area

Algorithm 5: Calculation of the reconfigured topology for Case II

Data: NG
Result: $NG*$
begin
  $NG* \leftarrow NG$
  for *each* $t_{s_i} \in \{t_{s_1}, ..., t_{s_n}\} \subseteq NT*$ **do**
    if $t_{p_{s_{i\,vuls}}}$ *are "easy-to-exploit"* **then**
      for *each* $t_{s_j} \in \{t_{s_1}, ..., t_{s_n}\}$ **do**
        if $t_{s_j} != t_{s_i}$ *and* $t_{s_{j_{vuls}}}$ *are "hard-to-exploit" and* $D_{t_{s_i}, t_{s_j}} <= CR_{t_{s_i}, t_{s_j}}$ **then**
          Add t_{s_j} into $G_{t_{s_i}}$
        end
      end
      Sort $G_{t_{s_i}}$ based on $H_{t_{s_k}}$ reversely where $t_{s_k} \in G_{t_{s_i}}$
      for *each* $t_{s_k} \in G_{t_{s_i}}$ **do**
        Remove $es_{t_{ps_i}, t_{s_i}}$
        $t_{tp_{s_i}} \leftarrow t_{p_{s_i}}$
        $t_{p_{s_i}} \leftarrow t_{s_k}$
        Create $es_{t_{s_k}, t_{s_i}}$
        if $R(t_{s_i}, t_{bs})$ *is True and* $L(t_{s_i}, t_{bs}, l)$ *is True* **then**
          $H_{t_{s_i}} \leftarrow H_{t_{s_k}} + 1$
          $F(t_{s_i})$
          break
        else
          Remove $es_{t_{s_k}, t_{s_i}}$
          $t_{p_{s_i}} \leftarrow t_{tp_{s_i}}$
          Recover $es_{t_{ps_i}, t_{s_i}}$
        end
      end
    end
  end
end

with 360 meters * 360 meters (0.1296 square kilometres). Some sensor nodes are attached with the street lights or traffic lights for power supply (e.g., the sensor node connected to the base station) while other sensor nodes are attached to the bus stops or outside of the buildings. The average distance between the sensor nodes in the simulation area is 25 meters. The ZigBee specification indicates that the transmission range of ZigBee is 10 meters to 100 meters depending on the power output of the individual device and the surrounding environment. We use

an average communication range of 75 meters in our simulations. We assume the Controller software implemented and tested in [46] is used in our simulations to determine the routing policies. The Controller is installed on a remote server and controls the sensor network via the base station. After deployment, each sensor node communicates with its neighbours and chooses the closest node towards the base station in its communication range as the parent node. After the route discovery to the Controller, sensor nodes form a tree topology.

We use the attacker model specified in Section 4.1.3. In the simulations, we choose a sensor node which is located at the farthest distance with the base station as the entry point of the attacker. The attacker exploits this node to compromise other nodes as the stepping stones towards the base station. The attacker's capabilities are modelled via the security metrics, including attack success probability (e.g., the location of the attacker, the skills required by the attacker to exploit the vulnerabilities), attack cost (e.g., the tools used by the attacker), mean-time-to-compromise (i.e., the mean time to exploit the vulnerabilities).

We use several metrics to analyse the security and the performance of the networks before and after reconfiguration. In particular, we use the attack success probability, attack cost, mean-time-to-compromise, mean-attack-path-length, return-on-attacks to analyse the attacker's effort and gain; we use the attack impact and risk to analyse the severity of the potential attack. All security metrics are defined in Section 3.2. We also introduce the average shortest path length (*ASPL*) to analyse the network performance. It is used to compute the average number of hops for all nodes in the network to forward data towards the base station.

To provide the system information for the IoT Generator, we estimate the following metric values for the node vulnerabilities: the attack success probability, attack impact, attack cost and compromise rate. In Case I, we denote the buffer overflow vulnerability in the noise sensor node as v_{noise1} and the vulnerability of no verification of updates as v_{noise2}; we denote the buffer overflow vulnerability in the weather application of the weather and air quality

sensor node as v_{wthr1} and the buffer overflow vulnerability in the air quality application as v_{air1}, respectively. As the tools used by the attacker to exploit the vulnerabilities are easy to obtain (e.g., Metasploit [1]), we use low attack cost. For v_{noise2}, we extract the values of the attack impact and attack success probability from the CVSS base score [2]. The other three buffer overflow vulnerabilities allow the attacker to remotely exploit without any authentication. For v_{noise1} and v_{wthr1}, we assume medium access complexity (e.g., require the attacker with high skill) and the attacker gains full control of the node; thus, we use medium attack success probability and maximum attack impact; we estimate the compromise rate as once per week as the attacker needs to be close to the sensor network. For v_{air1}, we assume low access complexity and the attacker has partial impact of confidentiality, integrity and availability; thus, we use high attack success probability and medium impact value; we estimate the compromise rate as twice per week as the attacker needs to be close to the sensor network and the vulnerability is easy to exploit. We present the estimated metric values and the calculated values for the risk and return-on-attacks in Table 4.2.

Table 4.2: Metric values of node vulnerabilities in Case I.

Metric \ Vulnerability	v_{noise1}	v_{noise2}	v_{wthr1}	v_{air1}
asp_v	0.6	0.55	0.6	1.0
aim_v	10.0	9.5	10.0	6.4
ac_v	4.0	4.0	4.0	4.0
cr_v	0.006	0.006	0.006	0.012
r_v	6.0	5.2	6.0	6.4
roa_v	1.5	1.3	1.5	1.6

In Case II, we denote the buffer overflow vulnerability in the noise sensor node as v_{noise3} and the buffer overflow vulnerability in the weather sensor node as v_{wthr2}. As the tools used by the attacker to exploit vulnerabilities are easy to obtain, we use low attack cost. For both vulnerabilities, the attacker is able to remotely exploit the vulnerabilities without any authentication and gains full control after the node is compromised. We use the maximum impact value. We

assume v_{noise3} requires the attacker with high skill and v_{wthr2} requires the attacker with low skill; thus, we use medium attack success probability for v_{noise3} and high attack success probability for v_{wthr2}. To exploit either of the vulnerabilities, the attacker needs to be close to the sensor network. As v_{wthr2} is easy to exploit and v_{noise3} is hard to exploit, we estimate the compromise rate as twice per week for v_{wthr2} and once per week for v_{noise3}. We present the estimated metric values and the calculated values for the risk and return-on-attacks in Table 4.3.

Table 4.3: Metric values of node vulnerabilities in Case II.

Metric \ Vulnerability	v_{noise3}	v_{wthr2}
asp_v	0.6	0.9
aim_v	10.0	10.0
ac_v	4.0	4.0
cr_v	0.006	0.012
r_v	6.0	9.0
roa_v	1.5	2.3

In both cases, we vary the number of noise sensor nodes ranging from 10 to 90 with the increment of 10 in each simulation. Topology reconfiguration needs to satisfy the wireless communication range of the sensor node. When using the heuristic method to reconfigure the topology, we specify the number of increased hops from any node to the base station in the reconfigured network to be no more than 1, 2 and 3. In Case I, we assume the vendor releases the patch for the buffer overflow vulnerability in the noise sensor node. Thus, we compare the security and the performance of both the initial network and the reconfigured network after the buffer overflow vulnerability is fixed in the noise sensor nodes.

We use Akaroa2 [97] as the simulation environment to validate the simulation results. Akaroa2 is a tool for improving the credibility of results from quantitative stochastic simulation using automated sequential analysis. We pass the values of metrics as the parameters to Akaroa2. Akaroa2 collects a series of observations of the parameters and calculates a global estimate of each parameter. The simulation stops until Akaroa2 produces the mean value of the parameter. Therefore, the run

time of the simulation is determined by Akaroa2.

4.2.2 Simulation Steps

We develop the topology reconfiguration module and integrate it with the framework described in Section 3.2. Figure 4.1 shows the modified framework.

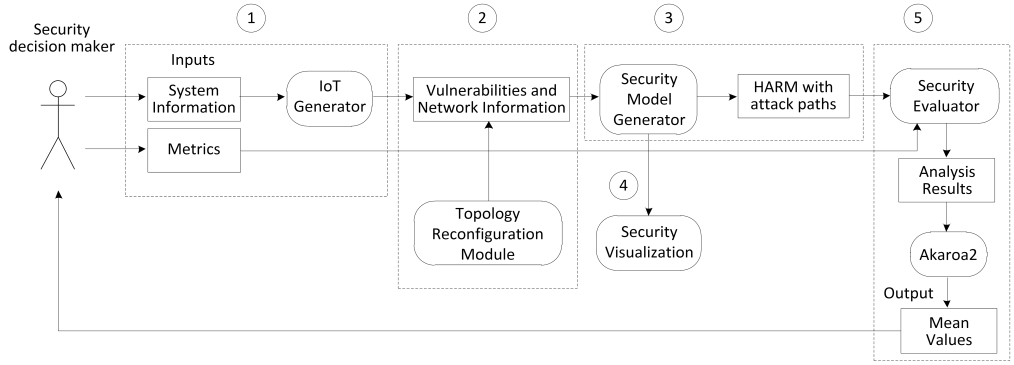


Figure 4.1: Framework with the topology reconfiguration module.

The framework has five phases: 1) data processing, 2) network reconfiguration, 3) security model generation, 4) security visualisation, 5) analysis and simulation. We use the framework to carry out the simulations. In each simulation, we randomly generate a network based on the network information and the node vulnerability information and also provide the metrics used for the further analysis (phase 1). We use the reconfiguration algorithm in the reconfiguration module to change the topology of the initial network (phase 2). Afterwards, we compute the graphical security models (i.e., the three-layered HARMs) for the two networks before and after reconfiguration (phase 3). Then, we evaluate the security and performance of both networks via the Security Evaluator along with the pre-determined metrics (phase 5). The results of each simulation are fed into the Akaroa2 which outputs the global estimate of each metric (phase 5).

We compare the security and performance of the networks before and after reconfiguration based on the mean values of the metrics calculated by Akaroa2. We use the percentage change to compare the mean metric values of the

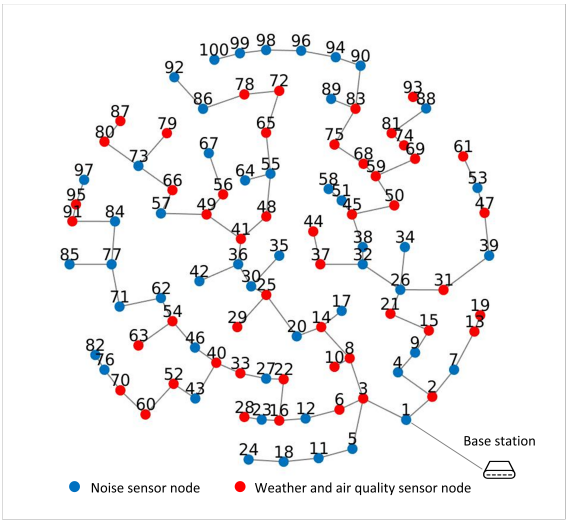
reconfigured network with the mean values of the initial network. We denote the percentage change as P_{change}, the mean metric value of the initial network as $Value_{ini}$ and the mean metric value of the reconfigured network as $Value_{rec}$. We calculate the percentage change by Equation (4.1). It represents the change of the mean metric value of the reconfigured network as a percentage of the value of the initial network. A negative percentage value indicates the metric value of the reconfigured network decreases while a positive percentage value indicates the metric value of the reconfigured network increases. We use the absolute value $|P_{change}|$ to decide the minimum and maximum of percentage change. We also calculate the mean value of the percentage change.

$$P_{change} = [(Value_{rec} - Value_{ini})/Value_{ini}] * 100 \tag{4.1}$$

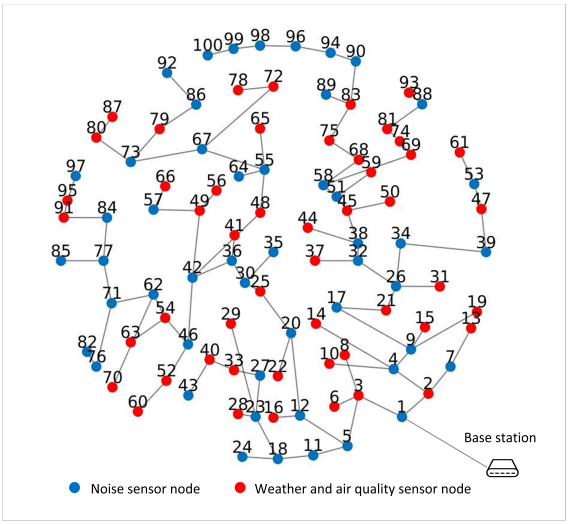
4.2.3 Security Modelling and Analysis of an Example Network

We use one example network to demonstrate the steps of security modelling and analysis in the simulation. We use Case I in the example network. The example network has a total number of 100 nodes with 50 noise sensor nodes and 50 weather and air quality sensor nodes. We randomly generate a network with the tree topology and use the optimal method to reconfigure the network topology. The topologies of the example network before and after reconfiguration are shown in Figure 4.2. Here, a blue circle represents a noise sensor node and a red circle represents a weather and air quality sensor node.

We denote the network before reconfiguration as the initial network (s_{ini}) and the network after reconfiguration as the optimal network as the optimal method is used (s_{opt}). An attacker is able to exploit the sensor node $t_{s_{95}}$ in the communication range as the entry point. We construct the three-layered HARMs for both networks, respectively. We visualise the attack paths captured in the HARMs in Figure 4.3. In both networks, the attacker is able to compromise some sensor nodes as stepping stones to eventually reach the node which is directly



(a) Before reconfiguration.

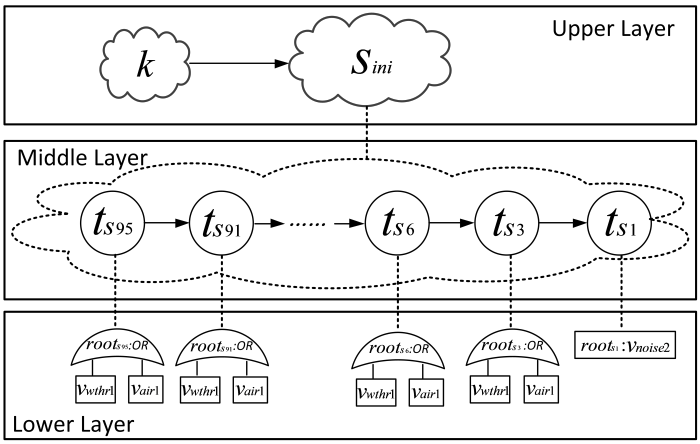


(b) After reconfiguration.

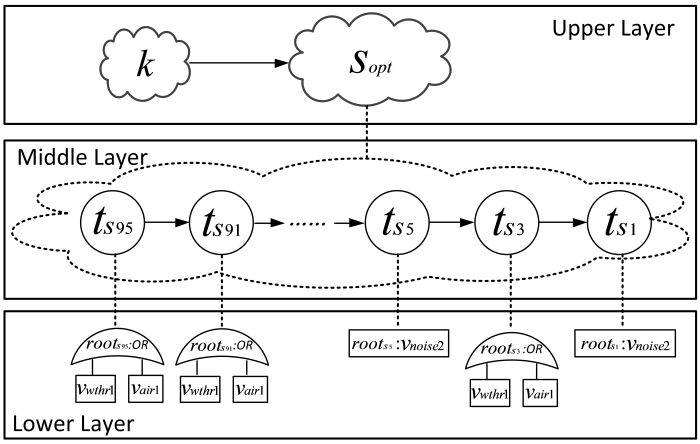
Figure 4.2: Topologies of the example network before and after reconfiguration.

connected to the base station.

We calculate the security and performance metrics using the Security Evaluator and show the results of the analysis in Table 4.4. As the optimal method has the restriction of no hop change, *MAPL* and *ASPL* do not change. Compared with the value in the initial network, *AC* does not change in the optimal network. The reasons are explained in the following. At first, all vulnerabilities used in Case I have the same attack cost. Secondly, the attacker can exploit either v_{noise1} or v_{noise2} in the noise sensor node. Therefore, after patching v_{noise1} in the



(a) Attack path in the HARM of the initial network.



(b) Attack path in the HARM of the optimal network.

Figure 4.3: Attack paths in HARMs of the example network.

optimal network, the attack cost values calculated from the vulnerability and then node levels do not change. As *MAPL* does not change, *AC* remains the same. *ASP* decreases while *MTTC* increases, which indicates more effort required by the attacker to reach the target in the optimal network. *ROA* decreases, which demonstrates the less gain of the attacker in the optimal network. Besides, the attacker has less impact and causes less potential harm on the optimal network as *AIM* and *R* decrease.

Table 4.4: Values of metrics of the initial network and optimal network.

Metric \ Value	Initial network	Optimal network
ASP	0.00837	0.00077
AIM	166.0	164.0
AC	68.0	68.0
MTTC	2083.3	2416.7
R	99.4	94.7
ROA	24.9	23.7
MAPL	17.0	17.0
ASPL	9.4	9.4

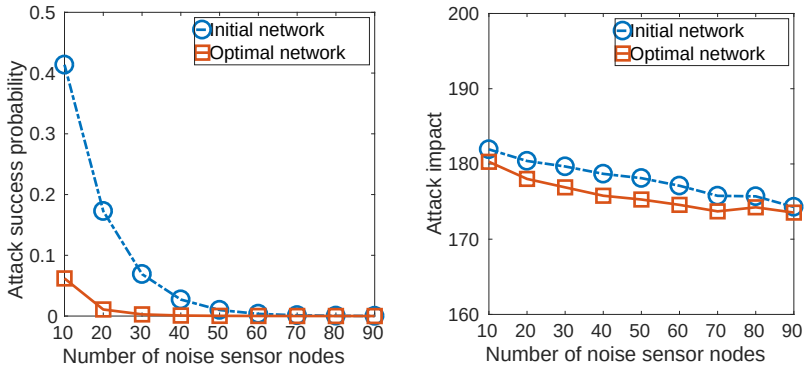
4.2.4 Simulation Results using Optimal Method

We compute the mean values of the metrics by Akaroa2 and compare the metrics before and after reconfiguration by Equation (4.1) in Section 4.2.2. We present and discuss the results using the optimal method without hop change in this section.

4.2.4.1 Results of Case I

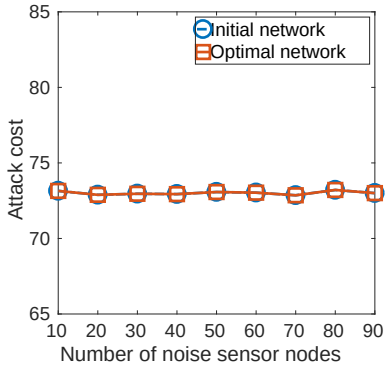
As the optimal method does not allow hop change in the reconfiguration, values of *MAPL* and *ASPL* in the optimal network remain the same as the values in the initial network. We plot the results of *ASP*, *AIM*, *AC*, *MTTC*, *R* and *ROA* in Figure 4.4.

Compared with the values of the metrics in the initial network, *ASP*, *AIM*, *R* and *ROA* decrease and *MTTC* increases in the optimal network for all different numbers of noise sensor nodes. With the increasing number of noise sensor nodes, *ASP*, *AIM*, *R* and *ROA* decrease and *MTTC* increases, because the noise sensor node has less number of vulnerabilities than the weather and air quality sensor node after patch. As the two types of nodes have the same attack cost for their vulnerabilities and the length of the attack path does not change, *AC* remains the same. We present the percentage changes of *ASP*, *AIM*, *MTTC*, *R* and *ROA* in Table 4.5. Minimum, maximum and mean percentage changes are calculated.

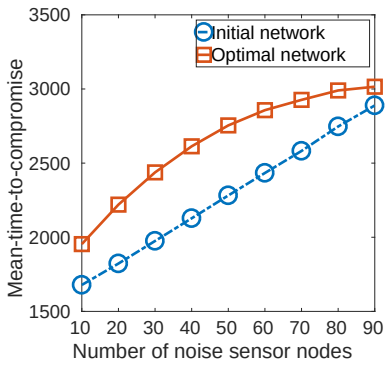


(a) *ASP*.

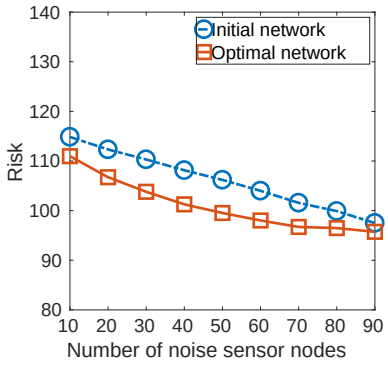
(b) *AIM*.



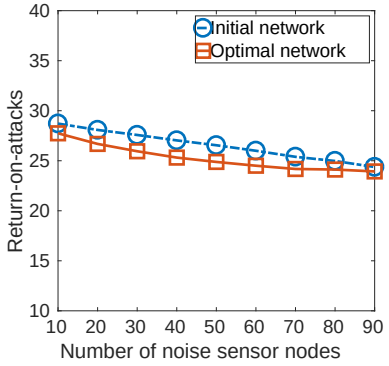
(c) *AC*.



(d) *MTTC*.



(e) *R*.



(f) *ROA*.

Figure 4.4: Mean values of metrics in Case I using optimal method.

Attacker's effort and gain: in the optimal network, *ASP* decreases significantly and *MTTC* increases moderately based on the mean percentage changes. This indicates the attacker needs to put more effort on compromising the target in terms of the access complexity and mean compromise time. *ROA* decreases slightly, which shows the attacker has less gain in the optimal network.

Table 4.5: Percentage changes of metrics in Case I using optimal method.

Percentage change / Metric	Minimum	Maximum	Mean
ASP	**-61.99%**	**-96.88%**	**-89.09%**
AIM	-0.43%	-1.63%	-1.21%
MTTC	4.38%	23.42%	**16.57%**
R	-1.81%	-6.35%	-4.74%
ROA	-1.81%	-6.35%	-4.74%

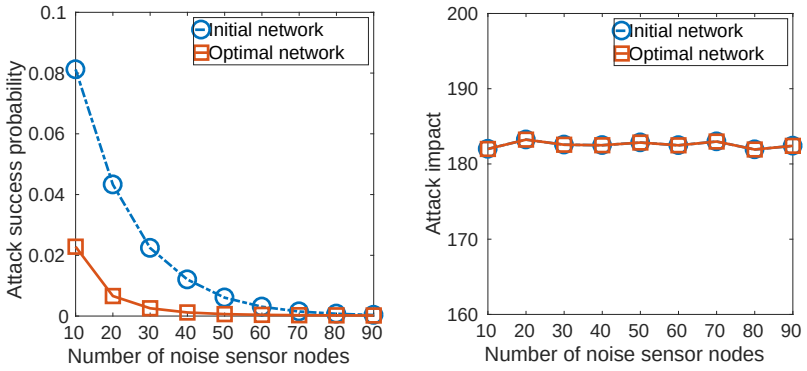
Severity of the attack: in the optimal network, *R* decreases slightly and *AIM* does not have much change, indicating that the attacker causes less potential harm in the optimal network while the impact remains similar.

4.2.4.2 Results of Case II

We plot the results of *ASP*, *AIM*, *AC*, *MTTC*, *R* and *ROA* in Figure 4.5. Compared with the values of the metrics in the initial network, *ASP*, *R* and *ROA* decrease and *MTTC* increase in the optimal network for all different numbers of noise sensor nodes. With the increasing number of noise sensor nodes, *ASP*, *R* and *ROA* decrease and *MTTC* increases, because the vulnerability of the noise sensor node is harder to exploit than the vulnerability of the weather and air quality sensor node. As the two types of nodes have the same attack cost and attack impact for their vulnerabilities and the length of the attack path does not change, *AC* and *AIM* remain the same. We present the percentage changes of *ASP*, *MTTC*, *R* and *ROA* in Table 4.6. Minimum, maximum and mean percentage changes are calculated.

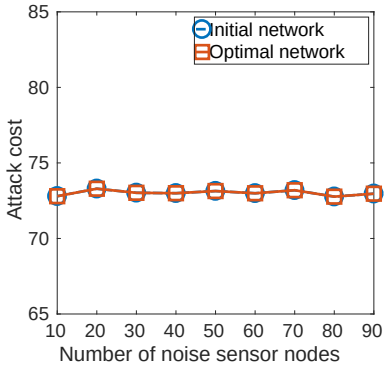
Attacker's effort and gain: in the optimal network, *ASP* decreases significantly and *MTTC* increases moderately based on the mean percentage changes. This indicates the attacker needs to put more effort on compromising the target in terms of the access complexity and mean compromise time. *ROA* decreases slightly, which shows the attacker has less gain in the optimal network.

Severity of the attack: *R* decreases slightly, indicating the less potential harm

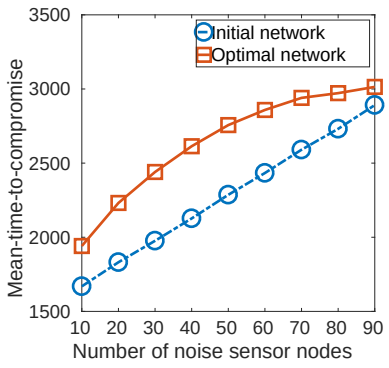


(a) *ASP*.

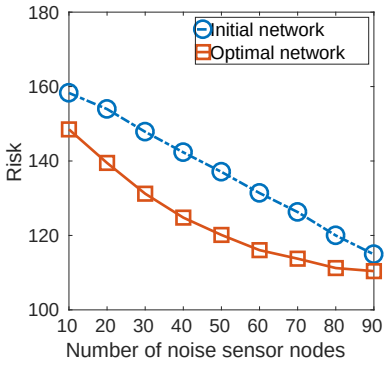
(b) *AIM*.



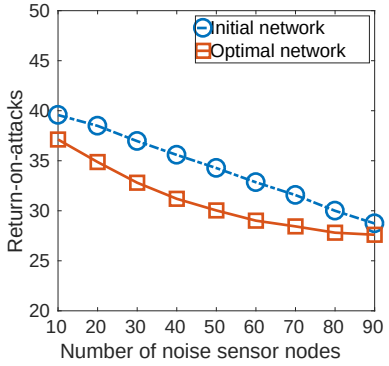
(c) *AC*.



(d) *MTTC*.



(e) *R*.



(f) *ROA*.

Figure 4.5: Mean values of metrics in Case II using optimal method.

caused by the attacker in the optimal network.

4.2.4.3 Summary

In both cases, by using the optimal method, *ASP* has significant decrease and *MTTC* has moderate increase. The largest change of *ASP* appears when the

Table 4.6: Percentage changes of metrics in Case II using optimal method.

Percentage change / Metric	Minimum	Maximum	Mean
ASP	**-45.62%**	**-90.15%**	**-78.64%**
MTTC	4.24%	23.49%	**16.57%**
R	-3.93%	-12.36%	-9.37%
ROA	-3.93%	-12.36%	-9.37%

number of noise sensor nodes is within 30 to 50. The performance of the optimal network remains the same as the initial network because the reconfiguration algorithm does not change the hop count. Overall, the optimal reconfiguration algorithm can effectively increase the attacker's effort in terms of the access complexity and mean compromise time and maintain the average shortest path length of the network.

4.2.5 Simulation Results using Heuristic Method

We present and discuss the results using the heuristic method with the consideration of three reconfiguration limitations. We denote the network before reconfiguration as the initial network and the network after reconfiguration as the reconfigured network with the specific reconfiguration limitation.

4.2.5.1 Results of Case I

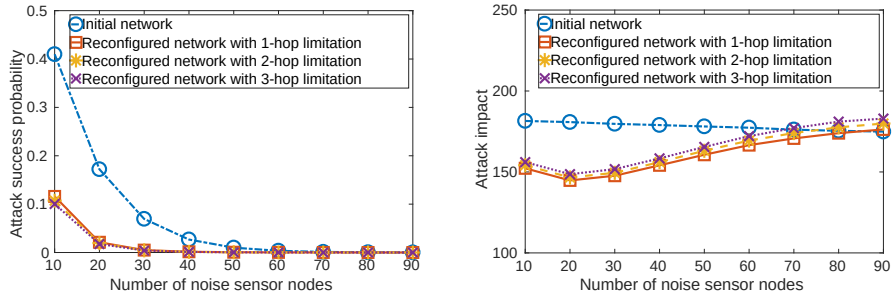
We plot the results of *ASP*, *AIM*, *AC*, *MTTC*, *R*, *ROA*, *MAPL* and *ASPL* in Figure 4.6. Compared with the values of the metrics in the initial network, metrics in the reconfigured networks with different reconfiguration limitations have the following changes: *ASP* decreases for all different numbers of noise sensor nodes, *MTTC* remains similar initially and then increases, *AIM*, *AC*, *MAPL* and *ASPL* decrease at first and then increase slightly while *R* and *ROA* decrease at first and then remain similar. With the increasing number of noise sensor nodes, for the reconfigured networks, *ASP* decreases, *MTTC* increases while values of other

metrics decrease before the number of noise sensor nodes is 20 and then increase steadily. When the number of noise sensor nodes is small, the sensor nodes try to connect to patchable parent nodes in their communication range. This can lead to several nodes connecting to the same parent node which may give them a path with shorter length to the base station. When the number of noise sensor nodes is very large, there is not much reconfiguration as the majority of nodes already connect to patchable noise sensor nodes. Thus, in the reconfigured networks, *MAPL* and *ASPL* decrease when the number of noise sensor nodes is very small and then increase steadily when the majority of the nodes are noise sensor nodes. We present the percentage changes of all metrics for the reconfigured networks with three reconfiguration limitations compared with the initial network in Table 4.7. Minimum, maximum and mean percentage changes are calculated.

Attacker's effort and gain: in the reconfigured networks, based on the mean percentage changes, *ASP* decreases significantly and *MTTC* increases moderately for all different hop limitations. This indicates the attacker needs to put more effort on compromising the target in terms of the access complexity and mean compromise time. *ROA* decreases moderately, which shows the attacker has less gain in the reconfigured networks. However, as the heuristic reconfiguration algorithm does not maximise the length of the path towards the base station, *MAPL* and *AC* decrease slightly, indicating that the attacker compromises less number of nodes to reach the target and spend less cost in the reconfigured networks.

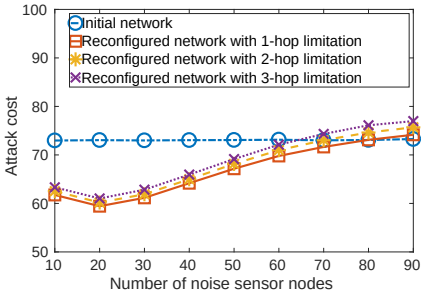
Severity of the attack: in the reconfigured networks, *R* decreases moderately and *AIM* decreases slightly for all different hop limitations, indicating that the attacker causes less potential harm and impact.

Performance: in the reconfigured networks with 1-hop and 2-hop limitations, *ASPL* decreases slightly, which shows a slight improvement of the performance. *ASPL* in the reconfigured network with 3-hop limitation remains similar as the value in the initial network.

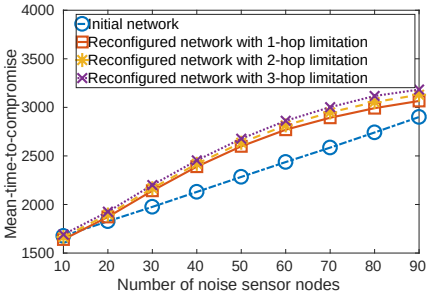


(a) *ASP*.

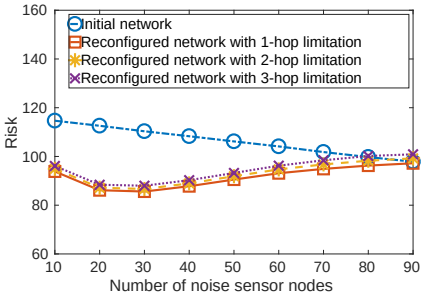
(b) *AIM*.



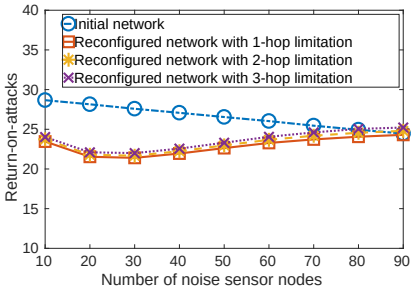
(c) *AC*.



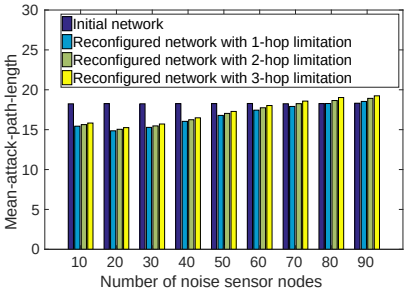
(d) *MTTC*.



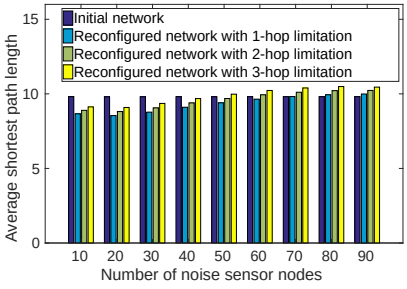
(e) *R*.



(f) *ROA*.



(g) *MAPL*.



(h) *ASPL*.

Figure 4.6: Mean values of metrics in Case I using heuristic method.

Table 4.7: Percentage changes of metrics in Case I using heuristic method.

Metric	Hop limitation	Percentage change		
		Minimum	Maximum	Mean
ASP	1	**-63.45%**	**-94.27%**	**-85.35%**
	2	**-68.81%**	**-94.85%**	**-87.19%**
	3	**-72.22%**	**-95.62%**	**-88.63%**
AIM	1	0.77%	-19.92%	-9.64%
	2	1.20%	-18.91%	-8.18%
	3	0.55%	-17.79%	-6.73%
AC	1	0.05%	-18.64%	-8.41%
	2	0.04%	-17.61%	-6.93%
	3	-1.40%	-16.46%	-5.45%
MTTC	1	-2.11%	13.81%	8.35%
	2	-0.59%	15.51%	**10.08**%
	3	0.92%	17.44%	**11.91**%
R	1	-0.65%	-23.49%	**-13.25%**
	2	1.41%	-22.58%	**-11.86%**
	3	0.43%	-21.49%	**-10.46%**
ROA	1	-0.65%	-23.49%	**-13.25%**
	2	1.41%	-22.58%	**-11.86%**
	3	0.43%	-21.49%	**-10.46%**
MAPL	1	0.05%	-18.64%	-8.41%
	2	0.04%	-17.61%	-6.93%
	3	-1.40%	-16.46%	-5.45%
ASPL	1	0.12%	-12.87%	-4.99%
	2	-1.20%	-10.09%	-2.20%
	3	-1.30%	-7.30%	0.59%

4.2.5.2 Results of Case II

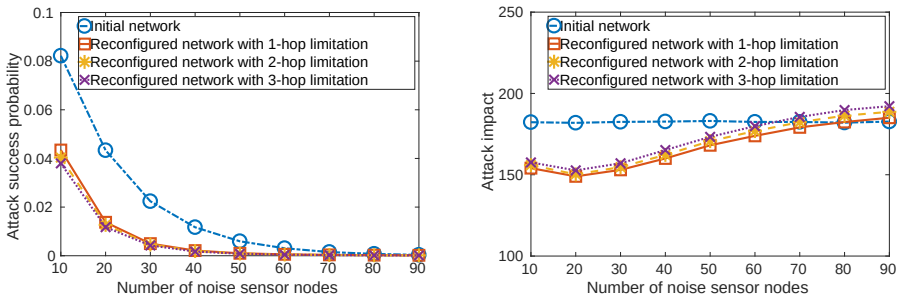
We plot the results of *ASP*, *AIM*, *AC*, *MTTC*, *R*, *ROA*, *MAPL* and *ASPL* in Figure 4.7. Compared with the values of the metrics in the initial network, metrics in the reconfigured networks with different reconfiguration limitations have the following changes: *ASP* decreases for all different numbers of noise sensor nodes, *MTTC* remains similar initially and then increases, *AIM*, *AC*, *MAPL* and *ASPL* decrease at first and then increase slightly while *R* and *ROA* decrease at first and then remain similar. With the increasing number of noise sensor nodes, for the reconfigured networks, *ASP* decreases, *MTTC* increases, *AIM*, *AC*, *MAPL* and *ASPL* decrease before the number of noise sensor nodes is

20 and then increase steadily while *R* and *ROA* decrease before the number of noise sensor nodes is 30 and then increase slowly. When the number of noise sensor nodes is small, the sensor nodes try to connect to parent nodes with the "hard-to-exploit" vulnerability in their communication range. This can lead to several nodes connecting to the same parent node which may give them a path with shorter length to the base station. When the number of noise sensor nodes is very large, there is not much reconfiguration as the majority of nodes already connect to noise sensor nodes with the "hard-to-exploit" vulnerability. Thus, in the reconfigured networks, *MAPL* and *ASPL* decrease when the number of noise sensor nodes is very small and then increase steadily when the majority of the nodes are noise sensor nodes. We present the percentage changes of all metrics for the reconfigured networks with three reconfiguration limitations compared with the initial network in Table 4.8. Minimum, maximum and mean percentage changes are calculated.

Attacker's effort and gain: in the reconfigured networks, based on the mean percentage changes, *ASP* decreases significantly and *MTTC* increases moderately for all different hop limitations. This indicates the attacker needs to put more effort on compromising the target in terms of the access complexity and mean compromise time. *ROA* decreases moderately, which shows the attacker has less gain in the reconfigured networks. However, as the heuristic reconfiguration algorithm does not maximise the length of the path towards the base station, *MAPL* and *AC* decrease slightly, indicating that the attacker compromises less number of nodes to reach the target and spend less cost in the reconfigured networks.

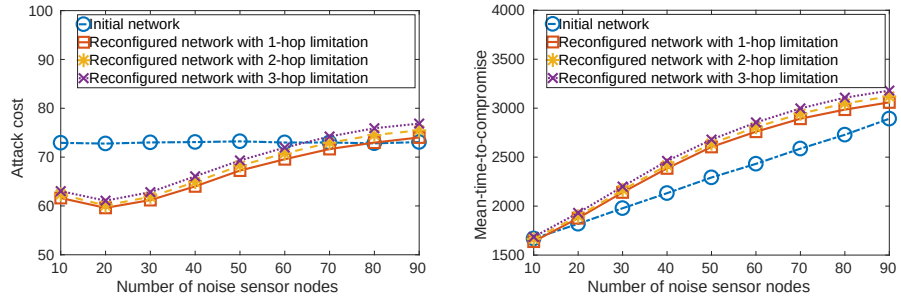
Severity of the attack: in the reconfigured networks, *R* decreases moderately and *AIM* decreases slightly for all different hop limitations, indicating that the attacker causes less potential harm and impact.

Performance: in the reconfigured networks with 1-hop and 2-hop limitations, *ASPL* decreases slightly, which shows a slight improvement of the performance. *ASPL* in the reconfigured network with 3-hop limitation remains similar as the



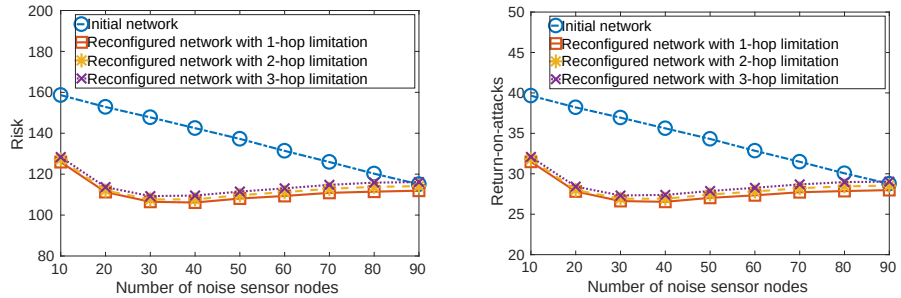
(a) *ASP*.

(b) *AIM*.



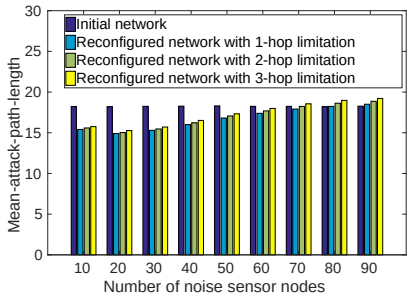
(c) *AC*.

(d) *MTTC*.

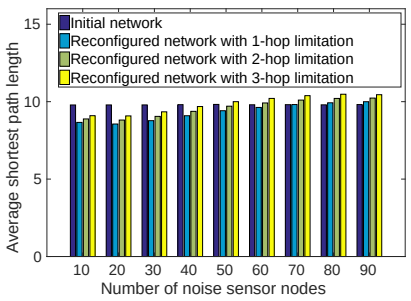


(e) *R*.

(f) *ROA*.



(g) *MAPL*.



(h) *ASPL*.

Figure 4.7: Mean values of metrics in Case II using heuristic method.

value in the initial network.

Table 4.8: Percentage changes of metrics in Case II using heuristic method.

Metric	Hop limitation	Percentage change		
		Minimum	Maximum	Mean
ASP	1	**-47.12%**	**-83.11%**	**-70.31%**
	2	**-50.92%**	**-84.66%**	**-73.28%**
	3	**-53.81%**	**-86.29%**	**-76.03%**
AIM	1	0.20%	-18.09%	-8.37%
	2	-0.08%	-17.41%	-6.96%
	3	-1.43%	-16.07%	-5.42%
AC	1	0.20%	-18.09%	-8.37%
	2	-0.08%	-17.41%	-6.96%
	3	-1.43%	-16.07%	-5.42%
MTTC	1	-1.82%	13.62%	8.40%
	2	-0.18%	15.42%	**10.08%**
	3	1.05%	17.39%	**11.98%**
R	1	-2.71%	-27.92%	**-17.92%**
	2	-0.79%	-27.15%	**-16.67%**
	3	0.94%	-26.06%	**-15.35%**
ROA	1	-2.71%	-27.92%	**-17.92%**
	2	-0.79%	-27.15%	**-16.67%**
	3	0.94%	-26.06%	**-15.35%**
MAPL	1	0.20%	-18.09%	-8.37%
	2	-0.08%	-17.41%	-6.96%
	3	-1.43%	-16.07%	-5.42%
ASPL	1	0.07%	-12.60%	-4.93%
	2	-1.16%	-9.98%	-2.19%
	3	-1.25%	-7.24%	0.58%

4.2.5.3 Summary

In both cases, by using the heuristic methods, *ASP* has significant decrease, *ROA* and *R* have moderate decrease and *MTTC* has moderate increase. The largest change of *ASP* appears when the number of noise sensor nodes is within 40 to 50.

In the reconfigured network with 3-hop limitation, for the percentage change, *ASP* has the largest decrease, *MTTC* has the largest increase, *MAPL* and *AC* have the lowest decrease while *AIM*, *R* and *ROA* have the lowest decrease, and *ASPL* does not have much change. Conversely, in the reconfigured network with 1-hop limitation, all security metrics have the opposite trends and *ASPL* has a slight

decrease. As the heuristic algorithms do not maximise the length of the path towards the base station, the changes of the metrics are due to both more number of patchable nodes or nodes with the "hard-to-exploit" vulnerability along the path and the shorter length of the path towards the base station. Therefore, among three different reconfiguration limitations of hops, the reconfiguration with 3-hop limitation makes the attacker have highest access complexity and longest mean compromise time, guarantees the lowest decrease of attack path length and attack cost, and maintains the average shortest path length; however, it also causes the lowest decrease of potential harm, impact and gain for the attacker.

Overall, the heuristic reconfiguration algorithms can effectively increase the attacker's effort in terms of the access complexity and mean compromise time, decrease the attacker's gain and the potential harm, and maintain the average shortest path length of the network.

Chapter 5

Optimal Defence Mechanisms for the IoT

Deception technology is a proactive approach in cyber defence [5]. It allows defenders to capture and analyse malicious behaviours by luring the attackers into the decoy systems within the networks and interacting with the attackers. As normal users do not know the existence of the decoy systems, defenders will only get alerts caused by malicious intrusions. It adds an extra layer of defence onto the traditional security solutions (e.g., Intrusion Detection System (IDS), firewalls, and endpoint anti-virus software). A honeypot is one of the commonly used technologies in cyber deception [5]. It is created as a fake asset and deployed around the valuable assets to divert attackers. However, the management complexity and scalability issues of the honeypots hinder the wide usage by the enterprises. Modern deception technology uses basic honeypot technology with the addition of automation technique [110]. It allows distributed deployment and update of decoy systems to achieve adequate coverage and remain cost-effective.

Security patches are used to fix vulnerabilities in the software to prevent systems from possible exploits thus decreasing the attack surface. However, the effective patch solution has been a big challenge for the IoT devices. First, many

manufacturers sell the IoT devices with no mechanism in place for automated patch updates. They are only in favour of usability but neglect security in the design phase [74] in order to gain quick access to the IoT market. This leaves the consumers with unsupported and vulnerable devices after purchase. Second, some IoT devices (e.g., medical devices) use commercial off-the-shelf (COTS) software for the operating system or runtime environment. Patches for the COTS may not be installed without the validation of the patches by the manufacturers. In order to patch the IoT devices, the enterprises need to have agreements with the manufacturers that clarify the obligations for creating and evaluating the patches for the devices and also testing the patches for the COTS running on the devices during the lifecycle support.

In this chapter, we propose a novel approach to compute the optimal deployments of two defence mechanisms for the IoT. The defence mechanisms are (1) the strategic deployment of adaptive deception technology and (2) the security patch management solution. We discuss related work on the selection of optimal defence mechanisms for the IoT and cyber deception technologies applied in the IoT, present the evaluation metrics along with the graphical security model for assessing the effectiveness and efficiency of the proposed defence mechanisms and the multi-objective optimisation problem for maximising security and minimising deployment cost, show a case study using an example network in the smart hospital to demonstrate the viability of the approach and perform simulations to validate the efficacy of the optimisation algorithm.

To the best of our knowledge, this work is the first approach to evaluate the combinations of deception technology and security patch solution and to compute the optimal deployments of these mechanisms for the IoT networks via the graphical security model. The main contributions are summarised as follows:

- Design evaluation metrics and apply the graphical security model to evaluate the combinations of the modern deception technology and patch

management solution for the IoT (Section 5.1.4);

- Compute the optimal deployments of the defence mechanisms for the IoT under the budget constraint via the multi-objective genetic algorithm (Sections 5.1.5 and 5.2.2);
- Provide the defenders with ways to compare and choose optimal deployments for the IoT (Section 5.2.3).

5.1 Optimisation Approach

We introduce the system model, defence mechanisms and attacker model followed by the formulation of the optimisation problem and optimisation steps.

5.1.1 System Model

We consider a general structure of the IoT network which consists of servers, client machines (e.g., computers) and IoT devices [53, 115]. We assume traditional defence mechanisms are already deployed on the IoT network, including IDSs, firewalls and anti-virus software on the servers and client machines. Additionally, the IoT network has a central patch management system to patch critical security vulnerabilities in the servers and client machines. Our focus is to deploy deception technology in the network and patch management solution for the IoT devices to defend against sophisticated attacks. We present an example network in Section 5.2.1.

5.1.2 Defence Mechanisms

We use two individual defence mechanisms (the strategic deployment of adaptive deception technology and the security patch management solution) and their combinations for the IoT.

5.1.2.1 Deception Technologies

Once attackers are inside the IoT network, they start probes to acquire information to determine the potential valuable assets and then move laterally in the network to launch attacks based on the information they gather during the probes [13]. In order to successfully lure the attackers, there are several problems to be considered, including where the decoy systems should be deployed, what types of decoys should be used and what level of authenticity of the decoys should be applied. In this section, we discuss these problems and the purchase (capital) cost associated with these deployments.

Modern deception technologies integrate honeypot technology, visualisation and automation technologies. There are several emerging vendors (e.g., Illusive Networks, Attivo Networks, TrapX, Cymmetria, TopSpin [110]) which implement modern deception technologies and also provide support for the IoT networks in various domains (e.g., smart home, smart office, healthcare).

There are two types of decoys/traps utilised throughout the network: emulation-based and full operating system (OS) based. Both emulation-based and full OS-based decoys can be autonomously created to fit within the environment with no changes to the existing IT infrastructure. They provide a variation of interactive capabilities. Emulated decoys allow the defenders to create a variety of fake assets (e.g., the IoT devices, endpoints, servers, routers) and to provide a large-scale coverage across the network. Full OS-based decoys allow replication of the actual production devices to increase the possibility of engagement with the attacker and to reveal the attacker's intention. In order to increase the overall decoy access, the combinations of the decoys should resemble the real usage of the network devices and guarantee the decoy diversity. In addition, decoys are suggested to be deployed in every VLAN of the network.

Deception technologies are implemented in different ways by different vendors [6, 10–12]. In general, the implementation usually consists of an intelligence centre and various types of decoys. The intelligence centre is used

to create, deploy and update a distributed decoy system, provide automated attack analysis, vulnerability assessment and forensic reporting, and integrate with other prevention systems (e.g., security incident and event management platform, firewalls) to block attacks. The decoy system includes the decoys that are deployed across the network. The intelligence centre can be purchased as a platform including hardware appliances and software. The decoys can be purchased individually with an annual license fee based on the number of servers, client machines and IoT devices to be protected.

5.1.2.2 Patch Management

The IoT network often consists of various types of the IoT devices that are produced by different manufacturers. We assume the enterprises have the annual maintenance contracts with these manufacturers for repair and upgrade services. However, there is no patch management solution for the IoT devices. Therefore, for each type of the IoT devices, the enterprises can make additional annual agreements with the manufacturers for the patch management solutions with a certain amount of investment. In the real world, there are more complex cases. For instance, different IoT devices produced by the same manufacturer can get the patch support in one contract; some manufacturers cannot provide patches for the IoT devices. However, this work aims to demonstrate the approach to select optimal deployments of defence mechanisms. More cases can be investigated in the future work.

5.1.3 Attacker Model

We assume an outsider attacker in the attacker model. The goal of the attacker is to steal private data from the IoT networks and sell it for economic gain. The assumptions of the attacker's capabilities are listed as follows.

- The ability to detect the deception depends on the knowledge gap between the attacker and reality. We assume a sophisticated attacker who has a high

probability to recognise the emulated decoy but may not be able to detect the full OS-based decoy.

- Once the attacker interacts with the decoy, his behaviour will be monitored. So if the attacker finds out the device is a decoy after interaction, he needs to terminate his actions immediately and find new ways to break into the network.

- The attacker is able to find exploitable but un-patched vulnerabilities or unknown vulnerabilities to compromise the servers and client machines [29].

- The attacker is able to exploit the vulnerabilities in the un-patched IoT devices via well-hidden malware (e.g., re-packing, polymorphism) and use them to compromise other devices (e.g., installation of backdoors for persistent attacks) [7].

5.1.4 Problem Formulation

We formulate the optimisation problem for the defence mechanisms in this section. We first introduce the deployment vector as the problem input and three evaluation metrics used in the evaluation of the deployments, followed by the definition of the optimisation problem.

We use the definitions of the IoT network and the graphical security model in Section 3.2 and add the description of decoy nodes. An IoT network can be defined as $IoT = (S, T, V)$ where S is a finite set of subnets, T is a finite set of nodes and V is a finite set of vulnerabilities. For each node $t \in T$, we add three attributes as follows.

- $t_{decoy} \in \{False, True\}$: indicate the node is either real or decoy.

- $t_{pr} \in (0, 1]$: demonstrate the probability that an attacker will interact with the node and use it as the stepping stone to compromise other nodes; if

the node is real, the attacker will have the probability of 1 to exploit the vulnerabilities and then use it to compromise other nodes; if the node is a decoy, the attacker may figure out the trap after interaction and then terminate his behaviour, or may not detect the trap and be diverted to other decoy nodes afterwards; we assume the attacker will not completely avoid the decoy, thus there is no zero probability.

- t_{cost}: show the deployment cost of the decoy (i.e., 0 if $t_{decoy} \equiv False$).

5.1.4.1 Deployment Vector

We assume the IoT network is divided into several VLANs (i.e., subnets). All decoy nodes deployed across the network have different types. Let Y_d denote the set of node types for deception deployment and Y_p denote the set of the IoT types for patch management. We define the deployment vector as follows.

Definition 1. The deployment vector dv is defined as an integer valued vector. The function $o : Y_d \rightarrow \{0, 1, 2\}$ describes the integer value for each type of decoy deployment in the network. The function $q : Y_p \rightarrow \{0, 1\}$ describes the integer value of the patch solution for each type of the IoT nodes in the network. Let dv_d denote the deployment vector for the deception technology and dv_p denote the deployment vector for the patch management. We denote the deployment vectors in the following:

- $dv_d = (o(type_1), o(type_2), ..., o(type_{|Y_d|}))$.
- $dv_p = (q(type_1), q(type_2), ..., q(type_{|Y_p|}))$.
- $dv = dv_d \cup dv_p$.

We assume at least one server decoy should be deployed in the network to engage with the attackers. Therefore, for the server decoy, the integer value indicates the specific type of server decoy is not deployed (0), emulated (1) or full OS-based (2). We use emulated decoys for the client machines and IoT devices.

So the integer value means that a specific type of decoy is either deployed (1) or not (0). For the patch management, the integer value shows that a specific type of the IoT devices can be either patched (1) or not patched (0).

5.1.4.2 Evaluation Metrics

We assume an attacker may have one or multiple targets in the network for achieving his goal (e.g., steal data stored in servers). For each target, the attacker may be able to find multiple attack paths to reach it via one or multiple entry points. We consider a set of attack paths AP for reaching all the targets from all possible entry points. We divide AP into two sets: AP_r to represent the set of attack paths with the real nodes as targets and AP_d to represent the set of attack paths with the decoy nodes as targets. As the attacker will terminate his behaviour once the trap is detected, AP_r only contains real nodes.

We introduce three new metrics (1) decoy node fraction (*DNF*), (2) node interaction probability (*NIP*), (3) the residual cost fraction (*RCF*) as the evaluation metrics. We also use three cost metrics for the deployment of deception technology and patch solution in the calculation of the residual cost fraction.

We propose *DNF* as the first evaluation metric. The decoy node fraction is defined as the average fraction of the decoy nodes among all nodes along the attack path towards the decoy target. Let ap_{dt} ($ap \in AP$) denote the set of decoy nodes along an attack path. We show the calculation of *DNF* by Equation (5.1).

$$DNF(IoT, dv) = \frac{\sum_{ap \in AP_d} \frac{|ap_{dt}|}{|ap|}}{|AP_d|} \tag{5.1}$$

We propose *NIP* as the second evaluation metric. The decoy interaction probability is defined as the average probability that an attacker will interact with the nodes along the attack path and eventually reach the decoy target. We show

the calculation of *NIP* by Equation (5.2).

$$NIP(IoT, dv) = \frac{\sum_{ap \in AP_d} \prod_{t \in ap} t_{pr}}{|AP_d|} \tag{5.2}$$

We propose *RCF* as the third evaluation metric. We first define the residual cost as the total cost of the deployment of all potential defence mechanisms subtracted by the cost of one deployment. Then the residual cost fraction is the residual cost divided by the total cost. To simplify the calculation of this metric, we assume the manufacturer charges the patch management solution for the same type of the IoT devices at a fixed price. Additionally, we assume all decoys are different to guarantee the diversity of the decoy system as identical decoys are suspicious to attackers.

We introduce three cost metrics used in the calculation of *RCF*: the cost for purchasing the intelligence centre as a platform denoted as *IC*, the total patch management cost for the IoT devices denoted as *PMC* and the deployment cost of all decoys denoted as *DC*. Let *TC* denote the total cost and pmc_{type} denote the patch solution cost for one type of the IoT devices. We show the calculations of *DC*, *PMC* and *RCF* by Equations (5.3), (5.4) and (5.5), respectively.

$$DC(IoT, dv) = \sum_{\substack{t_{decoy} \equiv True \\ t_{type} \in Y_d \\ o(t_{type}) > 0}} t_{cost} \tag{5.3}$$

$$PMC(IoT, dv) = \sum_{\substack{type \in Y_p \\ q(type) \equiv 1}} pmc_{type} \tag{5.4}$$

$$RCF(IoT, dv) = \frac{TC - (IC + DC + PMC)}{TC} \tag{5.5}$$

As the IoT devices are usually used as the entry points by the attacker, patching the IoT devices reduces the number of entry points and the number of attack paths towards the targets. Besides, the decoy node fraction and node interaction probability are attack path-based metrics. Therefore, the effect of patching can be expressed by the evaluation metrics.

5.1.4.3 Optimisation Problem

The optimisation problem is a multi-objective problem to maximise the decoy node fraction, node interaction probability and residual cost fraction defined by Equation (5.6).

$$\begin{aligned} &maximise \quad DNF(IoT, dv) \\ &maximise \quad NIP(IoT, dv) \\ &maximise \quad RCF(IoT, dv) \end{aligned} \tag{5.6}$$

Given that each deployment of the defence mechanisms entails purchase and maintenance cost, the optimisation problem is to compute a set of Pareto optimal points [92] that provides a reasonable balance between the effectiveness and efficiency of the deployments of the defence mechanisms. Let DV denote all possible deployments of the defence mechanisms for a given IoT network and $PP = \{DNF(IoT, dv), NIP(IoT, dv), RCF(IoT, dv) \mid dv \in DV\}$ denote all possible values of the evaluation metrics for the given DV. We define the Pareto optimal points in our optimisation problem as follows.

Definition 2. The Pareto optimal points POP for the three-objective optimisation problem are $\{(dnf^*, nip^*, rcf^*) \in PP\} \iff \nexists\, (dnf, nip, rcf) \in PP$ such that $dnf \geq dnf^* \wedge nip \geq nip^* \wedge rcf \geq rcf^*$, and $dnf > dnf^* \vee nip > nip^* \vee rcf > rcf^*$.

5.1.5 Optimisation Steps

We discuss the optimisation steps in this section. We modify the framework in Section 3.2 by adding the Deployment Generator, Deployment Evaluator and Optimisation Module. The new framework is shown in Figure 5.1 which consists of five phases: 1) data processing, 2) deployment generation, 3) security model generation, 4) deployment evaluation and 5) deployment optimisation.

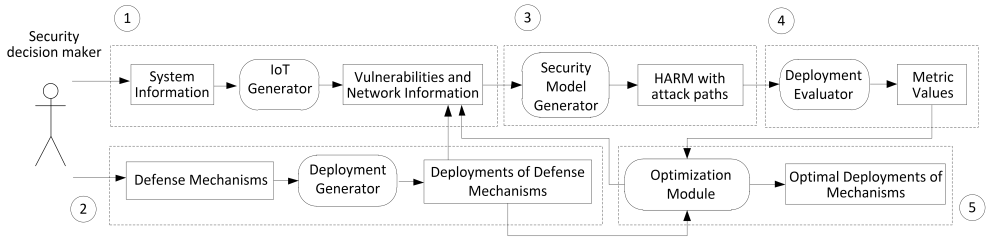


Figure 5.1: Framework with the optimisation steps.

In phase 1, the security decision maker provides the IoT Generator with the system information to construct an IoT network. The inputs include the network information and node vulnerability information.

In phase 2, given the network, we list all potential deployments of the defence mechanisms and represent them in the integer formats. The Deployment Generator randomly generates a set of different deployments. For each deployment of the defence mechanisms, we will feed it into the network and also pass it onto the Optimisation Module for further analysis.

In phase 3, we generate the three-layered HARM of the given IoT network with the deployment of the defence mechanisms. The Security Model Generator takes the constructed network as input and automatically generates the HARM in which all possible attack paths are captured in the model.

In phase 4, the Deployment Evaluator is used to evaluate the deployments of the defence mechanisms for the network. The three-layered HARM with the attack path information is taken as input into the Deployment Evaluator along with the determined evaluation metrics. The evaluation results are generated by

the Evaluator and fed into the Optimisation Module.

In phase 5, the optimal deployments are computed by the Optimisation Module. Based on the initial set of the deployments and the associated evaluation results, the Optimisation Module applies the multi-objective genetic algorithm to compute the optimal deployments of the defence mechanisms for the IoT network based on the termination conditions (e.g., the maximum number of generations defined by the security decision maker).

There are several reasons we select the genetic algorithm (GA) to solve the multi-objective optimisation problem. First, GA provides a simple way to encode the candidate solutions. As we use integer values to represent the deployment of the defence mechanisms, binary encoding can be easily applied to convert the integer values into binary values for our defence mechanisms. Second, due to the rapidly growing IoT network, scalability of the algorithm becomes a critical issue. However, GA requires little information to search effectively in a large search space which satisfies the requirement of an IoT network with a large number of nodes. Specifically, we choose one of the widely used GAs named NSGA-II in [40] as the optimisation algorithm. NSGA-II is defined as a fast sorting and elite multi-objective genetic algorithm and is able to find better spread of solutions.

5.2 Case Study

We use an example network in the smart hospital scenario. As a hospital system contains highly valuable information, attackers can have high economic gain for selling healthcare data in the black market. In this section, we introduce the example network and explain the steps for solving the optimisation problem via the proposed framework. Our approach is not limited to this specific case but applicable to the general IoT networks.

5.2.1 Example Network

We consider the picture archive and communication system (PACS) in a smart hospital network. The system consists of the PACS servers for the storage of image information from multiple source machine types, the PACS client machines to access the images and the Internet of Medical Things (IoMT) using radiology techniques [7, 8] to send images to servers. The PACS uses digital imaging and communications in medical (DICOM) standard as the communication protocol between the IoMT devices and the PACS servers. The example PACS network is shown in Figure 5.2.

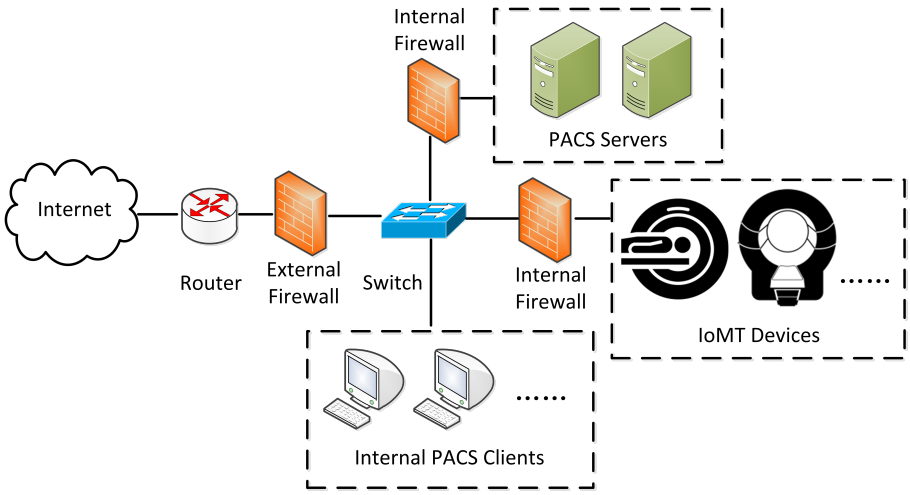


Figure 5.2: An example PACS network.

The PACS network is divided into three VLANs by the external and internal firewalls. The PACS servers use the active-active high availability cluster configuration to ensure reliable data storage and access. The redundant servers are identical in terms of both hardware and software. The PACS clients are computers with the DICOM viewers. The IoMT devices include Ultrasound machine, MRI machine, digital X-Ray machine and CT scanner. As these IoMT devices are closed systems, additional security software cannot be easily integrated into the devices. Besides, patches for the COTS running on the IoMT devices need to be verified and tested by the manufacturers due to safety issue.

We make assumptions for the operating systems (OSs) and applications running on the devices: each PACS server runs a Linux OS and is installed with the PACS software and database (e.g., MySQL); the PACS client machines run two types of OSs, including Windows 8 and Windows 10; the IoMT devices run Windows 7. All the chosen OSs and applications are commonly used in the PACS.

We use the attacker model described in Section 5.1.3. Specifically, the attacker is able to wrap highly capable attack tools into the obsolete malware (e.g., MS08-067), use it to exploit the vulnerabilities in the un-patched IoMT devices and become un-detected by the anti-virus software running on other devices [7, 8]. For the servers and client machines, the known but un-patched vulnerabilities can be collected from the National Vulnerability Database (NVD). We assume the attacker is able to exploit the client machines and IoMT devices as entry points, then move laterally in different VLANs of the network and eventually reach the servers for stealing private data.

The intelligence centre is purchased as a platform and decoys are purchased individually with an annual license fee. The IoMT devices are purchased from different manufacturers. The hospital has maintenance support from the manufacturers for the cleaning, repair and upgrade services but without the coverage of security updates. Thus the hospital needs to make additional contracts with the manufacturers separately for the patch management support. We investigate the prices of the deception products [131,132] and service fees for the IoMT devices [9, 86–88]. The estimated prices for the defence mechanisms are shown in Table 5.1.

5.2.2 Computation of the Optimal Deployments

We consider a small-scale PACS network, including 2 PACS servers, 10 PACS client machines (5 running Windows 8 and 5 running Windows 10) and 4 different IoMT devices. The network is divided into three VLANs: servers in VLAN1, client machines in VLAN2 and IoMT devices in VLAN3.

Table 5.1: Estimated prices for the defence mechanisms.

Product		Price (USD)
• Deception Deployment		
Intelligence Centre		20,000
PACS Server	Full OS (Linux)	1,500
	Emulation (Linux)	400
PACS Client (Emulation)	Windows 8	300
	Windows 10	300
IoMT Device (Emulation)	Ultrasound	200
	MRI	200
	X-Ray	200
	CT Scanner	200
• Patch Solution		
IoMT Device	Ultrasound	2,000
	MRI	6,000
	X-Ray	1,000
	CT Scanner	5,000

We assume each device has one known but un-patched vulnerability that can be exploited by the attacker to gain the root permission. Therefore, if the un-patched Windows 7 in the IoMT device is patched, there is no other exploitable vulnerability. We also assume each decoy has one vulnerability to lure the attacker. More vulnerabilities can be used based on the real configuration of the decoys. Besides, for the emulated decoy, the attacker has a probability of 0.5 to interact with it, exploit the vulnerability and then use it to compromise other devices; for the full OS-based decoy, this probability is 0.9. The probability values can be estimated based on the real configuration of the decoys.

Based on the concepts of GA [92], a population represents a group of potential deployments of the defence mechanisms; a chromosome corresponds to a deployment vector; a generation is one time of algorithm iteration; fitness values are determined by the evaluation metrics (i.e., fitness functions). We use the following algorithm parameters: population size = 100, maximum number of generations = 100, crossover rate = 0.8 and mutation rate = 0.2.

We first generate the PACS network via the IoT Generator. The network is represented as $IoT_{pacs} = (S_{pacs}, T_{pacs}, V_{pacs})$ where $S_{pacs} = \{s_{vlan1}, s_{vlan2}, s_{vlan3}\}$,

$T_{pacs} = \{t_{svr1}, t_{svr2}, t_{clt1}, ..., t_{clt10}, t_{iomt1}, ..., t_{iomt4}\}$ and $V_{pacs} = \{v_{svr1}, v_{svr2}, v_{clt1}, ..., v_{clt10}, v_{iomt1}, ..., v_{iomt4}\}$. We show a list of attributes (used in the optimisation problem) for t_{iomt1} as an example.

- $t_{iomt1_{type}} = Ultrasound\ (Win7)$
- $t_{iomt1_{decoy}} = False$
- $t_{iomt1_{pr}} = 1.0$
- $t_{iomt1_{cost}} = 0$

Given the network, we have the potential deployments of the defence mechanisms based on the types of devices as follows.

- $Y_d = \{PACS\ server\ (Linux), PACS\ client\ (Win8), PACS\ client\ (Win10), Ultrasound\ (Win7), MRI\ (Win7), XRay\ (Win7), CT\ (Win7)\}$
- $Y_p = \{Ultrasound\ (Win7), MRI\ (Win7), XRay\ (Win7), CT\ (Win7)\}$

Given the potential deployments, we randomly generate a population of different deployments DV_{pacs} ($|DV_{pacs}| = 100$). We show one example of the deployment vector (dv_1).

$$
\begin{aligned}
dv_{1d} =& (o(PACS\ server\ (Linux)), o(PACS\ client\ (Win8)), o(PACS\ client\ (Win10)), \\
& o(Ultrasound\ (Win7)), o(MRI\ (Win7)), o(XRay\ (Win7)), o(CT\ (Win7))) \\
=& (2,1,0,1,0,0,0) \\
dv_{1p} =& (q(Ultrasound\ (Win7)), q(MRI\ (Win7)), q(XRay\ (Win7)), q(CT\ (Win7))) \\
=& (1,0,1,1) \\
dv_1 =& dv_{1d} \cup dv_{1p} = (2,1,0,1,0,0,0,1,0,1,1)
\end{aligned}
$$

Afterwards, the PACS network is reconstructed with the defence mechanisms. For example, using dv_1, three decoy nodes are added into the network (t_{svrd1}, t_{cltd1} and t_{iomtd1}) and three IoMT devices are patched (t_{iomt1}, t_{iomt3} and t_{iomtd4}).

We show a list of attributes (used in the optimisation problem) for t_{iomtd1} as an example.

- $t_{iomtd1_{type}} = Ultrasound\ (Win7)$
- $t_{iomtd1_{decoy}} = True$
- $t_{iomtd1_{pr}} = 0.5$
- $t_{iomtd1_{cost}} = 200$

The reconstructed network is fed into the Security Model Generator to construct the three-layered HARM and to capture the potential attack paths. We show the attack paths in the HARM for the network with the deployment vector dv_1 in Figure 5.3.

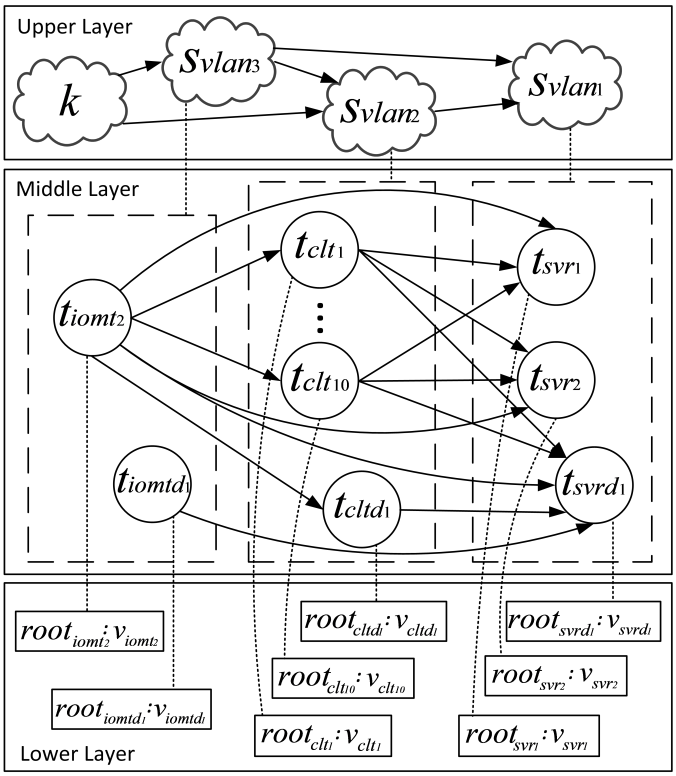


Figure 5.3: HARM for the PACS network with the deployment vector dv_1.

From Figure 5.3, the attacker is able to take the client machines in VLAN2 and IoMT devices in VLAN3 as entry points and then move laterally in the network

to eventually reach the servers. As decoys are also deployed in the network, the attacker may be lured into the decoys. Once the attacker interacts with the decoy, he either detects the decoy and terminates his behaviour, or is diverted to another decoy. We show the attack paths with the decoy servers as targets $AP_d = \{t_{iomt2} \rightarrow t_{cltd1} \rightarrow t_{svrd1}, t_{iomt2} \rightarrow t_{svrd1}, t_{iomt2} \rightarrow t_{clt1}/.../t_{clt10} \rightarrow t_{svrd1}, t_{iomtd1} \rightarrow t_{svrd1}, t_{clt1}/.../t_{clt10} \rightarrow t_{svrd1}, t_{cltd1} \rightarrow t_{svrd1}\}$.

The three-layered HARM with the attack path information is taken as input into the Security Evaluator to evaluate the deployment of the defence mechanisms. We show the results of the evaluation metrics for the network with dv_1 as an example.

$$
\begin{aligned}
DNF(IoT_{pacs}, dv_1) &= \frac{\sum_{ap \in AP_d} \frac{|ap_{dt}|}{|ap|}}{|AP_d|} \approx 0.47917 \\
NIP(IoT_{pacs}, dv_1) &= \frac{\sum_{ap \in AP_d} \prod_{t \in ap} t_{pr}}{|AP_d|} \approx 0.84375 \\
DC(IoT_{pacs}, dv_1) &= t_{svrd1_{cost}} + t_{cltd1_{cost}} + t_{iomtd1_{cost}} \\
&= 1500 + 300 + 200 = 2000 \\
PMC(IoT_{pacs}, dv_1) &= pmc_{Ultrasound(Win7)} + pmc_{XRay(Win7)} + pmc_{CT(Win7)} \\
&= 2000 + 1000 + 5000 = 8000 \\
RCF(IoT_{pacs}, dv_1) &= \frac{TC - (IC + DC + PMC)}{TC} \\
&= \frac{36900 - (20000 + 2000 + 8000)}{36900} \approx 0.18699
\end{aligned}
$$

The Optimisation Module takes the initial population of the deployments (DV_{pacs}) and the corresponding fitness values (PP_{pacs}) as inputs and then computes the optimal deployments via the multi-objective GA. We show the final population of the fitness values in Figure 5.4 which forms the Pareto optimal points (POP_{pacs}).

There are four labelled points in Figure 5.4 which represent the deployments with one maximum fitness value respectively. We show the deployment vector

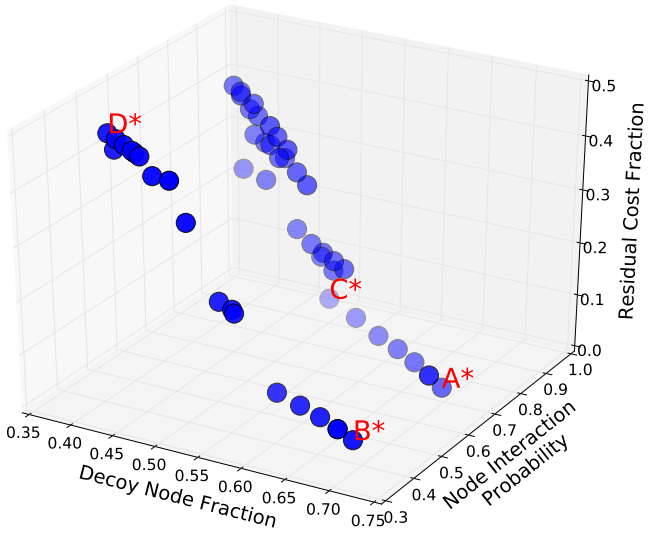


Figure 5.4: Final population of the deployments.

for each point as follows:

- $A*$: $dv_{A*} = (2,1,1,1,1,1,1,1,1,1,1)$
- $B*$: $dv_{B*} = (1,1,1,1,1,1,1,1,1,1,1)$
- $C*$: $dv_{C*} = (2,0,0,0,0,0,0,1,1,1,1)$
- $D*$: $dv_{D*} = (1,0,0,0,0,0,0,0,0,0,0)$

Points $A*$ and $B*$ have the maximum decoy node fraction. In these two deployments, all decoys are deployed and all IoMT devices are patched. The only difference is that point $A*$ represents the deployment with the full OS-based server decoy and point $B*$ represents the one with the emulated server decoy. Points $C*$ has the maximum node interaction probability and represents the deployment with the full OS-based server decoy deployed and all the IoMT devices patched. Point $D*$ has the maximum residual cost fraction and represents the deployment with one emulated server decoy deployed. These deployments will be used to test the algorithm accuracy in Section 5.3.

5.2.3 Analysis and Comparison of the Defence Mechanisms

In this section, we analyse the optimal deployments of the defence mechanisms and compare the optimal deployments with the deployments of the individual defence mechanisms. We introduce several metrics which are used in the following analysis: the percentage of decoys among all real devices (*PD*), percentage of patched devices among all real devices (*PPD*), number of attack paths towards the real targets (*NAPRT* calculated by $|AP_r|$), number of attack paths towards the decoy targets (*NAPDT* calculated by $|AP_d|$) and deployment cost of the defence mechanisms (*DCDM*).

5.2.3.1 Analysis of the Optimal Deployments

We assume the hospital has a budget for the security investment and use the budget and total cost to calculate the minimum residual cost fraction. Here, we use a budget of 25000 USD and calculate the minimum *RCF* which is approximately 0.322. We show the deployments which satisfy the budget constraint in Figure 5.5 (i.e., the points that are above 0.322 in Figure 5.4).

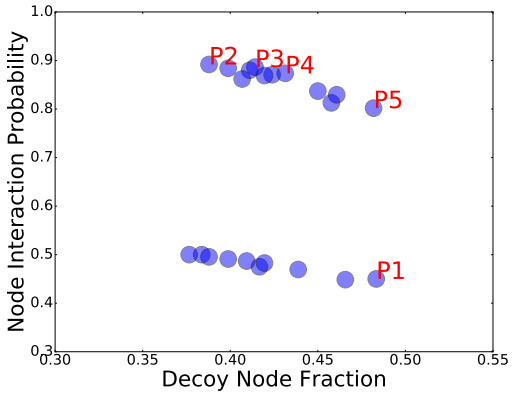


Figure 5.5: Deployments which satisfy the budget constraint.

In order to balance the effect of the two evaluation metrics in Figure 5.5, we introduce two weight values for the two metrics, denoted as β and γ, in which β represents the importance of *DNF* and γ represents the importance of *NIP*.

We calculate the weighted metric by $\beta DNF + \gamma NIP$ ($\beta + \gamma = 1$). We use the weight value of β ranging from 0 to 1 (with the increment of 0.1) and show the points with the maximum weighted metric in Figure 5.5. We summarise the corresponding deployment vector and the weight values of β for each point in the following:

- P_1: $dv_{P_1} = (1,1,1,0,1,1,0,1,0,1,0)$; $\beta = 1$
- P_2: $dv_{P_2} = (2,0,0,1,0,0,0,0,0,0,0)$; $\beta = 0$, 0.1
- P_3: $dv_{P_3} = (2,0,0,0,0,0,1,1,0,1,0)$; $\beta = 0.2$, 0.3, 04
- P_4: $dv_{P_4} = (2,0,0,1,0,0,1,1,0,1,0)$; $\beta = 0.5$
- P_5: $dv_{P_5} = (2,1,1,1,1,1,1,0,0,1,0)$; $\beta = 0.6$, 0.7, 0.8, 0.9

Point P_1 represents the deployment with the maximum decoy node fraction. Other points represent the deployments with one or multiple maximum weight metric values. We calculate the decoy percentage *PD* and patch percentage *PPD* for each point to indicate the coverage of the defence mechanisms. We show the values of two evaluation metrics (*DNF* and *NIP*) and percentage values for each point in Table 5.2 (in the order of the increasing weight value of β).

Table 5.2: Comparisons among the optimal deployments.

Point	*DNF*	*NIP*	*PD*	*PPD*
P_2	0.388	0.892	12.5%	0.0%
P_3	0.414	0.886	12.5%	12.5%
P_4	0.431	0.874	18.8%	12.5%
P_5	0.482	0.802	43.8%	6.3%
P_1	0.483	0.450	31.3%	12.5%

From point P_2 to point P_4 in Table 5.2, *DNF*, *PD* and *PPD* increase while *NIP* decreases with the increasing value of β, which indicates the higher percentages of the decoys and the patched IoMT devices, and the higher decoy authenticity (deployment of the full OS-based server decoy) when the importance of *DNF* increases. However, values of *PD* for the three points are much lower than

the values of points P_5 and P_1. Point P_5 has the maximum *PD*, relatively high values of *DNF* and *NIP* and a low value of *PPD*, which demonstrates the maximum percentage of the decoys, high decoy authenticity but a low percentage of the patched IoMT devices. Point P_1 has the maximum *DNF* and *PPD*, a low value of *NIP* and a relatively high value of *PD*, which demonstrates the high percentages of the decoys and the patched IoMT devices but a low decoy authenticity (deployment of all emulated decoys).

Among the points P_1, P_2, P_3 and P_4, we can see that there is a balance between the decoy percentage and decoy authenticity. Point P_1 has a good percentage of the decoys (more decoys to trap the attackers) but low authenticity of the decoys (lower probability to interact with the attackers once they are diverted to the server decoy) while the other three points have the opposite effect. Point P_5 achieves a good percentage of the decoy and high authenticity of the decoys but has a low percentage of the patched IoMT devices. Besides, points P_1, P_3 and P_4 have a good patch percentage. In order to facilitate the decision making on the optimal deployment of the defence mechanisms by the defenders, we summarise the analysis results in the following:

- Choose the deployment using dv_{P_5} to achieve high decoy coverage and decoy authenticity;
- Choose the deployment using dv_{P_1} to achieve high decoy coverage and patch coverage;
- Choose the deployment using dv_{P_4} to achieve high decoy authenticity and patch coverage.

5.2.3.2 Comparison of the Defence Mechanisms

We compare the three optimal deployments of the two defence mechanisms with the deployments of only deception or only patch solution in Table 5.3 using *PD*, *PPD*, *NAPRT*, *NAPDT* and *DCDM*. Additionally, we patch all the IoMT

devices by using only patch solution and deploy the full OS-based server decoy and all the other potential decoys by using only deception mechanism.

Table 5.3: Comparisons among the deployments of the defence mechanisms.

Metric / Defence	*PD*	*PPD*	*NAPRT*	*NAPDT*	*DCDM*
No defence	0.0%	0.0%	108	0	0
Only patch	0.0%	25.0%	20	0	14000
Only deception	43.8%	0.0%	108	68	22900
Both with dv_{P_1}	31.3%	12.5%	64	40	24400
Both with dv_{P_4}	18.8%	12.5%	64	34	24900
Both with dv_{P_5}	43.8%	6.3%	86	55	23900

From Table 5.3, compared with no defence, different combinations of the defence mechanisms increase the security of the IoT network at different aspects. The deployment of patch solution has the highest patch percentage and lowest number of real attack paths. However, once the attacker breaks into the network, the real devices are the only targets and the behaviour of sophisticated attackers may not be detected. The deployment of deception has the highest decoy percentage and highest number of fake attack paths but remains the highest number of real attack paths. The deployments of both patch and deception decrease the number of real attack paths significantly and introduce the fake attack paths to divert the attacks from the real assets. Therefore, the combinations of the two defence mechanisms are more effective to increase the security of the IoT networks with a reasonable cost.

5.3 Simulations

We compare the GA with the exhaustive search algorithm (ESA) in terms of time efficiency and accuracy of results. Here the accuracy refers to the ratio between the number of deployments with the maximum fitness value of one fitness function in the final population using the GA and that in the Pareto optimal points using the ESA. All simulations are performed using the computer equipped

with a 3.4 GHz CPU under Linux Mint 18.1 Serena and Eclipse Neon.1 with Python 2.7.

We consider the IoT networks with the similar structure of the example network used in the case study along with the attacker model in Section 5.1.3. We use the network with a fixed number of servers and client machines and an increasing number of the IoT devices with different types to investigate the impact of growing IoT devices on the proposed approach. Specifically, we use 2 servers, 50 client machines with two types of OSs. The number of the IoT devices ranges from 50 to 200 with an increment of 25 in each simulation. Servers, client machines and different types of the IoT devices are deployed in different VLANs. We assume every 25 IoT devices have the same type and belong to the same manufacturer. Each manufacturer makes agreement with the enterprise to implement patch solutions for the IoT devices at a different price (ranging from 1000 to 4000 with a difference of 500 for each agreement). Prices for the deception products are the same as the prices shown in Table 5.1. We use the following algorithm parameters for the simulations: population size = 100, maximum number of generations = 100, crossover rate = 0.8 and mutation rate = 0.2.

We show the time comparison of two algorithms in Figure 5.6. We use the number of nodes with different types (in the form of server-client machine-IoT) along with the number of bits used for the binary encodings of the deployments as labels. Initially, when the scale of the network is small, the ESA runs faster than the GA. However, with the increasing number of encoding bits, the time of the ESA increases exponentially while the time of the GA increases linearly.

We show the accuracy ratios of the GA compared with the ESA in Table 5.4. When the number of encoding bits ranges from 8 to 14, the accuracy ratio is 1.0 as the set of deployments with one maximum fitness value calculated by the GA is equal to the set of deployments with one maximum value calculated by the ESA. The ratio decreases when the number of encoding bits is equal to or larger than 16. We increase the population size and the maximum generation of GA and

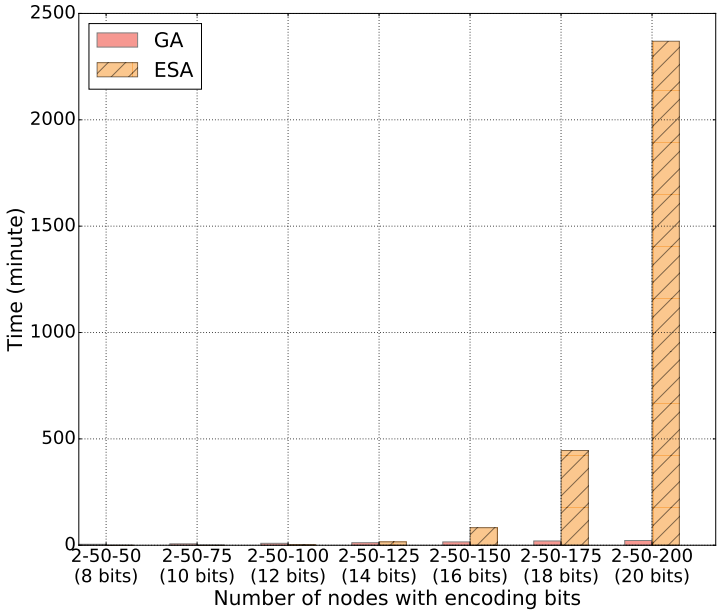


Figure 5.6: Runtime comparison of the GA and the ESA.

run the simulations with the same networks. We obtain the ratios of 1.0 for all these networks with a slightly higher runtime.

Table 5.4: Accuracy ratios of GA.

Network	Population size	Maximum generation	Runtime (minute)	Accuracy ratio
2-50-50 (8)	100	100	4	1.0
2-50-75 (10)	100	100	7	1.0
2-50-100 (12)	100	100	9	1.0
2-50-125 (14)	100	100	12	1.0
2-50-150 (16)	100	100	16	0.75
	100	**150**	**22**	**1.0**
2-50-175 (18)	100	100	20	0.75
	150	**150**	**40**	**1.0**
2-50-200 (20)	100	100	22	0.5
	150	**200**	**67**	**1.0**

In summary, we can conclude that the GA is much efficient than the ESA when the size of the IoT network increases and is able to obtain a good spread of the optimal deployments within a reasonable time.

Chapter 6

Discussions

The work presented in the thesis addresses all the research questions outlined in Chapter 1 by developing the security assessment framework for the IoT, developing the proactive defence mechanisms for the software-defined IoT with the non-patchable vulnerabilities, and developing the optimisation approach to compute the optimal deployments of the defence mechanisms. However, various assumptions have been made in the thesis that limit the scope of the research. In this chapter, we discuss the usability of the proposed approaches, address the limitations and point out possible extensions in the future work.

6.1 Usability

To address the research question Q1 in Section 1.2, we have developed the security assessment framework for the IoT to capture the potential attack scenarios and analyse the security of the IoT. In the security assessment framework, we have developed an IoT Generator, a Security Model Generator and a Security Evaluator. The IoT Generator creates an IoT network based on the subnet reachability information, node connectivity for each subnet and vulnerability information for each node; the Security Model Generator constructs the three-layered HARM based on the given IoT network to discover all possible

attack paths; the Security Evaluator analyses the security of the network using various security metrics. If different defence mechanisms are deployed in the IoT networks (both in device-level and network-level), the framework is able to calculate the security metrics for these mechanisms for comparing the effectiveness among them. Thus, the impact of various potential attacks can be mitigated by choosing and deploying the most effective defence mechanisms.

When using the communication protocols as the subnet classification method, some IoT devices with multiple communication modules provide connections among different subnets. These vulnerable cross-protocol devices create additional attack paths traversing different subnets and lead to unpredictable attacks. Therefore, the IoT devices that connect different subnets and extend the attack paths can be identified by the three-layered HARM. Potential countermeasures could be deployed on the cross-protocol IoT devices to prevent the spread of the attacks. In the three use cases for evaluating the framework, all the constructed IoT networks use the real-world vulnerability information (e.g., CVE or existing literature) and the topology information based on the assumptions of the real networks. Therefore, when applying the framework to the real-world IoT networks, the inputs will not have much difference compared with the system information used in the evaluation. Moreover, the framework is developed for the IoT but can also be used for the general computer networks. Based on the unique features of the two types of the networks, there will be different system models (i.e., many IoT devices use wireless communication protocols and the communication ranges of the devices affect the topology construction; the vulnerability information of the IoT devices is harder to obtain compared with that of the hosts in the computer networks as there are very few vulnerability scanners for the IoT), attacker models (e.g., attackers are easier to compromise the IoT devices as many of them are deployed in the open field) and defence mechanisms (e.g., traditional defence mechanisms may not be suitable for the IoT devices because of the limited computational power) provided to the framework. However, the procedures to compute the potential attack scenarios

and to analyse the security of the networks will be the same.

To address the research question Q2 in Section 1.2, we have utilised the SDN technique and developed two proactive defence mechanisms for the software-defined sensor networks in the IoT scenario with the non-patchable vulnerabilities. The idea behind the proposed mechanisms is to change the attack surface of the network in order to increase the attacker's efforts to compromise the target. We have considered two cases. In Case I, the network consists of a mix of the patchable and non-patchable nodes (i.e., nodes with only patchable vulnerabilities and nodes with only non-patchable vulnerabilities); the proactive defence mechanism is to maximise the number of the patchable nodes along the route to the base station. In Case II, the network has only non-patchable nodes; the defence mechanism is to maximise the number of nodes with the vulnerabilities which are harder to exploit along the route to the base station. From the simulations, the attacker's efforts to reach the target are increased when using the sensor nodes as the stepping stones after the topology reconfiguration.

As there are many papers on the implementation of the SDN solutions for the WSNs, we have applied the mechanisms onto the IoT-enabled sensor networks. The idea of the topology reconfiguration can be applied to other IoT-enabled networks when the SDN solutions are supported and implemented to achieve the flexible network management. Besides, the system model is constructed based on the real-world network deployment, device specification (e.g., the communication protocols), SDN implementation and vulnerability information. Therefore, the proposed mechanisms are applicable to the real-world IoT networks.

To address the research question Q3 in Section 1.2, we have developed the optimisation approach and considered two defence mechanisms: (1) the modern deception technology and (2) security patch solution. The patch solution is one of the traditional defence mechanisms and used to reduce the attack surface while the modern deception technology is a proactive defence mechanism that can effectively defend against the sophisticated attackers by luring them into the

decoys. The optimal deployments of the combinations of these two mechanisms are computed and compared via the optimisation approach to maximise the security of the IoT while minimising the deployment cost.

The proposed approach can be applied to any IoT networks with the potential deployments of these two defence mechanisms. Besides, the approach can be applied to decide the deployments of the decoys individually for the analysis of the modern deception technology on the IoT. Moreover, the computation of the optimal deployments of these two mechanisms based on the evaluation metrics and the comparisons among the individual and combined usages of the defence mechanisms provide insights on the investigation of combinations of other traditional and modern defence mechanisms.

6.2 Limitations and Future Work

There are several limitations in the thesis that can be investigated in the future research to extend the scope of the work.

Methodology: the methodology requires the collection and abstraction of the information from the systems and the real-world attacks followed by the proposed approaches and the framework. It aims to define the scope of the problem at first and then provide the solution. The computed results via the framework are affected by the estimated values of the inputs which may have some variation based on the semi-qualitative and semi-quantitative standard of the metric values.

Security assessment framework: we can use more scenarios as the use cases to evaluate the framework. Besides, the extensions could focus on the security analysis under different attacker models and framework validation shown in the following:

- More types of the attacker models could be considered in the security analysis of different use cases. For example, DDoS attacks target a single system using a large number of zombie computers (infected by

malwares). It is more disruptive for the IoT as the IoT devices can be easily compromised due to the limited security protections, and then controlled by the attackers as zombie devices [147]. Moreover, such attacks have already been carried out by the attackers in the real world. Thus analysing DDoS attacks via the graphical security model and finding the protection strategies to mitigate the impact of these attacks could be a promising research direction.

- Experiments could be performed in the real testbed to validate the framework. We could either design and set up a small-scale IoT system or use the existing testbeds [23, 99]. For example, we could design a smart sensing system consisting of various types of sensors (e.g., light sensor, sound sensor, ultrasonic range sensor, and temperature and humidity sensor) to monitor the environment, carry out attacks on the sensors, obtain the data from the testbed and feed it into the framework.

Securing the software-defined IoT: we can carry out simulations using different IoT scenarios (e.g., smart hospital, smart building) to evaluate the proactive defence mechanisms. Besides, the extensions could focus on the development of the reconfiguration algorithms and the design of the proactive defence mechanisms shown in the following:

- We have assumed that every IoT node in the network can be reconfigured. In some cases, critical devices cannot be reconfigured (e.g., sensors in the Body Sensor Network). We could consider more reconfiguration restrictions according to different networks in different scenarios to adapt the algorithms to the realistic cases.

- The current algorithms work for the nodes with one parent. We could release this restriction and design the algorithms for various network topologies (e.g., mesh, and cluster-tree).

- The optimal algorithm maximises the number of patchable nodes or nodes

with “hard-to-exploit” vulnerabilities along the path to the base station with the restriction of no changes of the hop counts. The heuristic algorithms release this restriction by introducing hop limitation after reconfiguration but do not maximise the number of patchable nodes or nodes with “hard-to-exploit” vulnerabilities along the path to the base station. We could develop algorithms that take consideration of maximising the number of patchable nodes or nodes with “hard-to-exploit” vulnerabilities along the path to the base station and the length of the path under the reconfiguration limitation of different hop changes.

- In the real world, the IoT devices may have a mix of the patchable and non-patchable vulnerabilities; non-patchable IoT devices may have a variety of vulnerabilities in which some are easy to exploit while others are hard to exploit. We could design more proactive defence mechanisms for these complex cases and implement the algorithms for the mechanisms.

- We have not considered the behaviour changes of the sensor nodes over time. The concept of ”Temporal-HARM (T-HARM)” has been introduced in [146] to capture the changes of the network at different time points and compute the graphical security model for the snapshot of the network at each time point. We could use the T-HARM to capture the behaviour changes of the sensor nodes and reconfigure the topology of the network when changes occur.

- We have used the average shortest path length to analyse the performance of the network after reconfiguration. However, the distances between different nodes also affect the performance of the sensor nodes. We could consider the response time as the performance metric to investigate the effectiveness of the mechanisms.

- When the network is compromised by the attacker, taking timely reactive actions is of vital importance. Based on the SDN functions, networks

can be reconfigured quickly and flexibly through the control plane. We could patch the non-compromised and patchable nodes to prevent potential attacks and isolate the compromised and non-patchable nodes when nodes are under attacks. We could also analyse the effectiveness of the reactive defence mechanisms with the support of the SDN via the framework.

Optimal defence mechanisms: we can use more case studies to analyse the effect of the defence mechanisms. Besides, the extensions could focus on the usage of the defence mechanisms and evaluation of the optimisation algorithms.

- We have used the annual license fee as the deployment cost of an individual decoy and the annual service fee as the deployment cost of the patch management solution for the IoT devices. We could consider the long-term effect of the annual fees on the deployment costs of both deception and security patch mechanisms.

- When analysing the optimal deployments of the defence mechanisms in the case study, we have assumed that each decoy is configured to have one vulnerability. We could use the real configurations of the decoys and make more realistic assumptions for the interaction probability values in the analysis.

- We have chosen the GA as the optimisation algorithm in the framework. We could evaluate other algorithms to find the most efficient algorithm for the optimisation problem.

Mobility of the IoT: we have assumed that the IoT topology is static throughout the thesis. However, one of the key features of the IoT is mobility. The movement of the heterogeneous devices has a great influence over the security of the IoT as the attack surface changes with the dynamically changing network. In different scenarios, the IoT devices have different movement patterns. Thus, a mobility model could be designed to capture the node movement in the network (e.g., nodes join or move out) and notify changes to other models

in the framework. In the wireless communication networks, mobility models have been designed and extensively used in the evaluations of the network protocols. In particular, a mobility model is used to describe the movement pattern of mobile objects and to represent changes of their location, velocity and acceleration over time [21]. The mobility models have been classified based on their characteristics, which include the random-based models [26, 32], models with temporal dependency or spatial dependency [25, 62], models with geographic constraints [70, 135], *etc*. The initial research has been done by one student in our lab. The work includes the discussion of current mobility models, integration of several mobility models with the graphical security model to capture the movement of the IoT devices regarding different scenarios and analysis of the impact of node movements on the IoT security. Future work could focus on the integration of different mobility models to be adapted to more realistic and complex IoT scenarios.

Chapter 7

Conclusions

Improving the security of the IoT is a difficult task as the IoT consists of a large number of heterogeneous and resource constrained devices with numerous exploitable vulnerabilities (some vulnerabilities may be non-patchable). In the thesis, we have made distinctive contributions to the knowledge in the domain of the IoT security modelling and assessment for mitigating the impact of the potential attacks targeting the IoT.

We have presented a security assessment framework for the IoT and provided a formal definition of the framework. We have also introduced three example networks in three different IoT scenarios, which are smart home, healthcare monitoring and environment sensing, and evaluated the framework via these scenarios. All possible attack paths have been computed by the three-layered HARM and the values of the chosen security metrics have been calculated in the security analysis phase. From the analysis results, the security decision maker is able to decide the most vulnerable part of the network, to assess the effectiveness of different defence mechanisms and to choose the most effective way to protect the network, thus mitigating the impact of the potential attacks.

In order to deal with the non-patchable vulnerabilities, we have considered two cases and presented two proactive defence mechanisms to change the attack surface of the software-defined sensor networks in the IoT scenarios. We have

developed both optimal and heuristic reconfiguration algorithms for the two cases to change the network topology according to the proactive defence mechanisms and integrated the algorithms into the framework. We have analysed the security and performance of the network via the graphical security model and various metrics. We have carried out simulations to evaluate our mechanisms. The simulation results have shown the mechanisms effectively increase the attack effort in terms of the attack success probability and mean-time-to-compromise and maintain the network performance in terms of the average shortest path length.

In order to improve the security of the IoT under the budget constraint, we have provided a novel approach to compute the optimal deployments of two defence mechanisms for the IoT networks. We have defined three evaluation metrics for assessing the effectiveness and efficiency of the deployments and formalised the optimisation problem. We have chosen the GA as the optimisation algorithm and integrated the algorithm and the evaluation metrics with the framework to carry out the computation of the optimal deployments of the defence mechanisms. We have shown the feasibility of the proposed approach via a case study. We have also performed simulations to compare the GA with the ESA in terms of runtime efficiency and result accuracy. Simulation results have demonstrated that the GA is time efficient and obtains a good spread of deployments.

Overall, the security assessment framework with the reconfiguration and optimisation modules is capable of discovering the potential attack paths in the IoT and evaluating the effectiveness of the defence mechanisms to mitigate the impact of potential attacks, changing the attack surface of the software-defined IoT with the non-patchable vulnerabilities to increase the attacker's efforts and optimising the deployments of the defence mechanisms under the budget constraint for the IoT.

References

[1] Metasploit: Penetration Testing Software. https://www.metasploit.com/. Last accessed: 2017-11-10.

[2] Vulnerability Details: CVE-2014-2378. http://www.cvedetails.com/cve/CVE-2014-2378/. Last accessed: 2017-09-27.

[3] IEEE Std 802.15.4-2003: Part 15.4: Wireless Medium Access Control and Physical Layer Specifications for Low-Rate Wireless Personal Area Networks. Technical report, IEEE, 2003.

[4] OpenFlow Switch Specification (Version 1.3.0). Technical report, ONF, 2012.

[5] Beginner's Guide To Cyber Deception. Technical report, vARMOUR, 2016.

[6] Know What is Lurking in Your Network. Technical report, Attivo, 2016.

[7] ANATOMY OF AN ATTACK MEDJACK (Medical Device Hijack). Technical report, TrapX Research Labs, 2017.

[8] ANATOMY OF AN ATTACK MEDJACK.2 Hospitals Under Siege. Technical report, TrapX Research Labs, 2017.

[9] How Much Does an Ultrasound Machine Cost? http://www.costowl.com/healthcare/healthcare-ultrasound-machine-costs.html, 2017. Last accessed: 2017-11-10.

[10] On the Radar: Attivo Networks offers deception, vulnerability assessment, and response automation. Technical report, Ovum, 2017.

[11] On the Radar: TopSpin adds IoT security capabilities to DECOYnet. Technical report, Ovum, 2017.

[12] Product Brief - DeceptionGrid 6.0. Technical report, TrapX Research Labs, 2017.

[13] Retail Point-of-Sale Under Fire. Technical report, TrapX Research Labs, 2017.

[14] H. Abie and I. Balasingham. Risk-based Adaptive Security for Smart IoT in eHealth. In *Proceedings of the 7th International Conference on Body Area Networks (BodyNets '12)*, pages 269–275. ICST, 2012.

[15] I. Agadakos, C. Y. Chen, M. Campanelli, P. Anantharaman, M. Hasan, B. Copos, T. Lepoint, M. Locasto, G. F. Ciocarlie, and U. Lindqvist. Jumping the Air Gap: Modeling Cyber-Physical Attack Paths in the Internet-of-Things. In *Proceedings of the 2017 Workshop on Cyber-Physical Systems Security and Privacy (CPS '17)*, pages 37–48. ACM, 2017.

[16] M. Albanese, S. Jajodia, and S. Noel. Time-efficient and cost-effective network hardening using attack graphs. In *Proceedings of the IEEE/IFIP International Conference on Dependable Systems and Networks (DSN '12)*, pages 1–12. IEEE, 2012.

[17] M. Anirudh, S. A. Thileeban, and D. J. Nallathambi. Use of honeypots for mitigating DoS attacks targeted on IoT networks. In *Proceedings of the 2017 International Conference on Computer, Communication and Signal Processing (ICCCSP '17)*, pages 1–4. IEEE, 2017.

[18] Q. M. Ashraf and M. H. Habaebi. Autonomic Schemes for Threat Mitigation in Internet of Things. *Journal of Network and Computer Applications*, 49(C):112–127, 2015.

[19] A. Atamli and A. Martin. Threat-Based Security Analysis for the Internet of Things. In *Proceedings of the 2014 International Workshop on Secure Internet of Things (SIoT '14)*, pages 35–43. IEEE Computer Society, 2014.

[20] Y. Bachy, F. Basse, V. Nicomette, E. Alata, M. Kaaniche, J. C. Courrege, and P. Lukjanenko. Smart-TV Security Analysis: Practical Experiments. In *Proceedings of the 2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN '15)*, pages 497–504. IEEE, 2015.

[21] F. Bai and A. Helmy. A Survey of Mobility Modeling and Analysis in Wireless Adhoc Networks. In *Wireless Ad Hoc and Sensor Networks*, chapter 1. Springer, 2006.

[22] Y. Berhanu, H. Abie, and M. Hamdi. A Testbed for Adaptive Security for IoT in eHealth. In *Proceedings of the International Workshop on Adaptive Security (ASPI '13)*, pages 1–8. ACM, 2013.

[23] M. Berman, J. S. Chase, L. Landweber, A. Nakao, M. Ott, D. Raychaudhuri, R. Ricci, and I. Seskar. GENI: A federated testbed for innovative network experiments. *Computer Networks*, 61(Supplement C):5–23, 2014.

[24] C. J. Bernardos, A. de la Oliva, P. Serrano, A. Banchs, L. M. Contreras, H. Jin, and J. C. Zuniga. An Architecture for Software Defined Wireless Networking. *IEEE Wireless Communications*, 21(3):52–61, 2014.

[25] C. Bettstetter. Smooth is Better Than Sharp: A Random Mobility Model for Simulation of Wireless Networks. In *Proceedings of the 4th ACM*

International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWIM '01), pages 19–27. ACM, 2001.

[26] C. Bettstetter, H. Hartenstein, and X. Pérez-Costa. Stochastic Properties of the Random Waypoint Mobility Model. *Wireless Networks*, 10(5):555–567, 2004.

[27] L. Bilge and T. Dumitras. Before We Knew It: An Empirical Study of Zero-day Attacks in the Real World. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security (CCS '12)*, pages 833–844. ACM, 2012.

[28] R. M. Blank and P. D. Gallagher. NIST Special Publication 800-30 Revision 1 Guide for Conducting Risk Assessments. Technical report, National Institute of Standards and Technology, 2012.

[29] D. Borbor, L. Wang, S. Jajodia, and A. Singhal. Securing Networks Against Unpatchable and Unknown Vulnerabilities Using Heterogeneous Hardening Options. In *Proceedings of IFIP Annual Conference on Data and Applications Security and Privacy (DBSec '17)*, pages 509–528. Springer International Publishing, 2017.

[30] J. Bort. REFRIGERATOR HACKED: Here's The Biggest Problem Facing The Internet Of Things. `http://www.businessinsider.com.au/hackers-use-a-refridgerator-to-attack-businesses-2014-1?r=US&IR=T`, 2014. Last accessed: 2017-12-01.

[31] P. Bull, R. Austin, E. Popov, M. Sharma, and R. Watson. Flow Based Security for IoT Devices Using an SDN Gateway. In *Proceedings of the 2016 IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud '16)*, pages 157–163. IEEE, 2016.

[32] T. Camp, J. Boleng, and V. Davies. A Survey of Mobility Models for Ad Hoc Network Research. *Wireless Communications and Mobile Computing*, 2(5):483–502, 2002.

[33] S. Chakrabarty, D. W. Engels, and S. Thathapudi. Black SDN for the Internet of Things. In *Proceedings of the 2015 IEEE 12th International Conference on Mobile Ad Hoc and Sensor Systems (MASS '15)*, pages 190–198. IEEE Computer Society, 2015.

[34] C. Y. Chang, C. H. Kuo, J. C. Chen, and T. C. Wang. Design and Implementation of an IoT Access Point for Smart Home. *Applied Sciences*, 5(4):1882–1903, 2015.

[35] P. Y. Chen, S. M. Cheng, and K. C. Chen. Information Fusion to Defend Intentional Attack in Internet of Things. *IEEE Internet of Things Journal*, 1(4):337–348, 2014.

[36] X. Chen, K. Makki, K. Yen, and N. Pissinou. Sensor Network Security: A Survey. *IEEE Communications Surveys & Tutorials*, 11(2):52–73, 2009.

[37] L. Coppolino, V. DAlessandro, S. DAntonio, L. Lev, and L. Romano. My Smart Home is Under Attack. In *Proceedings of the 2015 IEEE 18th International Conference on Computational Science and Engineering (CSE '15)*, pages 145–151. IEEE Computer Society, 2015.

[38] C. Cowan, F. Wagle, C. Pu, S. Beattie, and J. Walpole. Buffer overflows: attacks and defenses for the vulnerability of the decade. In *Proceedings of the DARPA Information Survivability Conference and Exposition (DISCEX '00)*, volume 2, pages 119–129, 2000.

[39] J. R. Crandall, Z. Su, S. F. Wu, and F. T. Chong. On Deriving Unknown Vulnerabilities from Zero-day Polymorphic and Metamorphic Worm Exploits. In *Proceedings of the 12th ACM Conference on Computer and Communications Security (CCS '05)*, pages 235–248, 2005.

[40] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[41] J. Deng, R. Han, and S. Mishra. Countermeasures Against Traffic Analysis Attacks in Wireless Sensor Networks. In *Proceedings of the 1st International Conference on Security and Privacy for Emerging Areas in Communications Networks (SecureComm '05)*, pages 113–126. IEEE, 2005.

[42] S. Dowling, M. Schukat, and H. Melvin. A ZigBee Honeypot to assess IoT Cyberattack Behaviour. In *Proceedings of the 2017 28th Irish Signals and Systems Conference (ISSC '17)*, pages 1–6. IEEE, 2017.

[43] A. El-Mougy, M. Ibnkahla, and L. Hegazy. Software-Defined Wireless Network Architectures for the Internet-of-Things. In *Proceedings of the 2015 40th Annual IEEE Conference on Local Computer networks (LCN '15)*, pages 804–811. IEEE Computer Society, 2015.

[44] L. Eschenauer and V. D. Gligor. A Key-Management Scheme for Distributed Sensor Networks. In *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS '02)*, pages 41–47. ACM, 2002.

[45] L. Gallon and J. Bascou. Using CVSS in Attack Graphs. In *Proceedings of the 6th International Conference on Availability, Reliability and Security (ARES '11)*, pages 59–66, 2011.

[46] L. Galluccio, S. Milardo, G. Morabito, and S. Palazzo. Sdn-wise: Design, prototyping and experimentation of a stateful sdn solution for wireless sensor networks. In *Proceedings of the 2015 IEEE Conference on Computer Communications (INFOCOM '15)*, pages 513–521. IEEE, 2015.

[47] A. D. Gante, M. Aslan, and A. Matrawy. Smart Wireless Sensor Network Management Based on Software-Defined Networking. In *Proceedings of the 2014 27th Biennial Symposium on Communications (QBSC '14)*, pages 71–75. IEEE, 2014.

[48] M. Ge, J. B. Hong, W. Guttmann, and D. S. Kim. A framework for automating security analysis of the internet of things. *Journal of Network and Computer Applications*, 83:12–27, 2017.

[49] M. Ge, J. B. Hong, S. E. Yusuf, and D. S. Kim. Proactive defense mechanisms for the software-defined Internet of Things with non-patchable vulnerabilities. *Future Generation Computer Systems*, 78(2):568–582, 2018.

[50] M. Ge and D. S. Kim. A Framework for Modeling and Assessing Security of the Internet of Things. In *Proceedings of the 2015 IEEE 21st International Conference on Parallel and Distributed Systems (ICPADS '15)*, pages 776–781. IEEE Computer Society, 2015.

[51] K. Gill, S. H. Yang, F. Yao, and X. Lu. A ZigBee-Based Home Automation System. *IEEE Transactions on Consumer Electronics*, 55(2):422–430, 2009.

[52] T. Goodspeed. MSP430 Buffer Overflow Exploit for Wireless Sensor Nodes. `http://travisgoodspeed.blogspot.co.nz/2007/08/machine-code-injection-for-wireless.html`. Last accessed: 2016-09-14.

[53] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami. Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions. *Future Generation Computer Systems*, 29(7):1645–1660, 2013.

[54] K. Habib and W. Leister. Threats Identification for the Smart Internet of Things in eHealth and Adaptive Security Countermeasures.

In *Proceedings of the 2015 7th International Conference on New Technologies, Mobility and Security (NTMS '15)*, pages 1–5. IEEE, 2015.

[55] A. Hakiri, P. Berthou, A. Gokhale, and S. Abdellatif. Publish/subscribe-enabled Software Defined Networking for Efficient and Scalable IoT Communications. *IEEE Communications Magazine*, 53(9):48–54, 2015.

[56] M. Hamdi and H. Abie. Game-Based Adaptive Security in the Internet of Things for eHealth. In *Proceedings of the 2014 IEEE International Conference on Communications (ICC '14)*, pages 920–925. ICST, 2014.

[57] P. S. Henry and H. Luo. WiFi: what's next? *IEEE Communications Magazine*, 40(12):66–72, 2002.

[58] J. Hong and D. Kim. HARMs: Hierarchical Attack Representation Models for Network Security Analysis. In *Proceedings of the 10th Australian Information Security Management Conference (AISM '12)*, 2012.

[59] J. B. Hong and D. S. Kim. Assessing the Effectiveness of Moving Target Defenses using Security Models. *IEEE Transactions on Dependable and Secure Computing*, 13(2):163–177, 2015.

[60] J. B. Hong and D. S. Kim. Towards Scalable Security Analysis using Multi-Layered Security Models. *Journal of Network and Computer Applications*, 75:156–168, 2016.

[61] J. B. Hong, D. S. Kim, C. J. Chung, and D. Huang. A survey on the usability and practical applications of Graphical Security Models. *Computer Science Review*, 26:1–16, 2017.

[62] X. Hong, M. Gerla, G. Pei, and C.-C. Chiang. A Group Mobility Model for Ad Hoc Wireless Networks. In *Proceedings of the 2nd ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM '99)*, pages 53–60. ACM, 1999.

[63] F. Hu, Q. Hao, and K. Bao. A Survey on Software-Defined Network and OpenFlow: From Concept to Implementation. *IEEE Communications Surveys Tutorials*, 16(4):2181–2206, 2014.

[64] X. Huang, P. Craig, H. Lin, and Z. Yan. SecIoT: a security framework for the Internet of Things. *Security and Communication Networks*, 2015.

[65] K. Ingols, M. Chu, R. Lippmann, S. Webster, and S. Boyer. Modeling Modern Network Attacks and Countermeasures Using Attack Graphs. In *Proceedings of the 25th Annual Computer Security Applications Conference (ACSAC '09)*, pages 117–126. IEEE Computer Society, 2009.

[66] K. Ingols, R. Lippmann, and K. Piwowarski. Practical Attack Graph Generation for Network Defense. In *Proceedings of the 22nd Annual Computer Security Applications Conference (ACSAC '06)*, pages 121–130. IEEE Computer Society, 2006.

[67] A. Jacobsson, M. Boldt, and B. Carlsson. A risk analysis of a smart home automation system. *Future Generation Computer Systems*, 56:719–733, 2016.

[68] S. Jha, O. Sheyner, and J. Wing. Two Formal Analys s of Attack Graphs. In *Proceedings of the 15th IEEE Workshop on Computer Security Foundations (CSFW '02)*, pages 49–63. IEEE Computer Society, 2002.

[69] F. Jia, J. B. Hong, and D. S. Kim. Towards Automated Generation and Visualization of Hierarchical Attack Representation Models. In *Proceedings of the 2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM '15)*, pages 1689–1696. IEEE, 2015.

[70] P. Johansson, T. Larsson, N. Hedman, B. Mielczarek, and M. Degermark. Scenario-based Performance Analysis of Routing Protocols for Mobile Ad-hoc Networks. In *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom '99)*, pages 195–206. ACM, 1999.

[71] S. D. T. Kelly, N. K. Suryadevara, and S. C. Mukhopadhyay. Towards the Implementation of IoT for Environmental Condition Monitoring in Homes. *IEEE Sensors Journal*, 13(10):3846–3853, 2013.

[72] P. Kinney. ZigBee Technology: Wireless Control that Simply Works. Technical report, ZigBee Alliance, 2003.

[73] J. Ko, C. Lu, M. B. Srivastava, J. A. Stankovic, A. Terzis, and M. Welsh. Wireless Sensor Networks for Healthcare. *Proceedings of the IEEE*, 98(11):1947–1960, 2010.

[74] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas. DDoS in the IoT: Mirai and Other Botnets. *Computer*, 50(7):80–84, 2017.

[75] B. Kordy, L. Piètre-Cambacédès, and P. Schweitzer. DAG-based attack and defense modeling: Don't miss the forest for the attack trees. *Computer Science Review*, 13-14:1–38, 2014.

[76] D. Kreutz, F. M. V. Ramos, P. E. Verìssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig. Software-Defined Networking: A Comprehensive Survey. *Proceedings of the IEEE*, 103(1):14–76, 2015.

[77] Q. D. La, T. Q. S. Quek, J. Lee, S. Jin, and H. Zhu. Deceptive Attack and Defense Game in Honeypot-Enabled Networks for the Internet of Things. *IEEE Internet of Things Journal*, 3(6):1025–1035, 2016.

[78] J. Lanza, L. Sánchez, L. Muñoz, J. A. Galache, P. Sotres, J. R. Santana, and V. Gutiérrez. Large-Scale Mobile Sensing Enabled Internet-of-things

Testbed for Smart City Services. *International Journal of Distributed Sensor Networks*, 11:157–157, 2015.

[79] B. Latré, B. Braem, I. Moerman, C. Blondia, and P. Demeester. A Survey on Wireless Body Area Networks. *Wireless Networks*, 17(1):1–18, 2011.

[80] B. Latre, B. Braem, I. Moerman, C. Blondia, E. Reusens, W. Joseph, and P. Demeester. A Low-delay Protocol for Multihop Wireless Body Area Networks. In *Proceedings of the 4th Annual International Conference on Mobile and Ubiquitous Systems: Networking Services (MobiQuitous '07)*, pages 1–8. IEEE, 2007.

[81] M. T. Lazarescu. Design of a WSN Platform for Long-Term Environmental Monitoring for IoT Applications. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 3(1):45–54, 2013.

[82] S. Lee, S. Kim, K. Choi, and T. Shon. Game theory-based Security Vulnerability Quantification for Social Internet of Things. *Future Generation Computer Systems*, 2017.

[83] T. Lei, Z. Lu, X. Wen, X. Zhao, and L. Wang. SWAN: An SDN Based Campus WLAN framework. In *Proceedings of the 2014 4th International Conference on Wireless Communications, Vehicular Technology, Information Theory and Aerospace Electronic Systems (VITAE '14)*, pages 1–5. IEEE, 2014.

[84] D. Leversage and E. James. Estimating a System's Mean Time-to-Compromise. *IEEE Security & Privacy*, 6(1):52–60, 2008.

[85] J. Liu, Y. Li, M. Chen, W. Dong, and D. Jin. Software-Defined Internet of Things for Smart Urban Sensing. *IEEE Communications Magazine*, 53(9):55–63, 2015.

[86] S. Loomis. MRI Service Cost Price Info. `https://info.blockimaging.com/mri-service-cost-price-info`, 2016. Last accessed: 2017-11-10.

[87] S. Loomis. X-Ray Equipment Service Cost Price Info. `https://info.blockimaging.com/x-ray-equipment-service-cost-price-info`, 2016. Last accessed: 2017-11-10.

[88] S. Loomis. CT Scanner Service Cost Price Info. `https://info.blockimaging.com/bid/95421/ct-scanner-service-cost-price-info`, 2017. Last accessed: 2017-11-10.

[89] T. Luo, H. P. Tan, and T. Quek. Sensor OpenFlow: Enabling Software-Defined Wireless Sensor Networks. *IEEE Communications Letters*, 16(11):1896–1899, 2012.

[90] A. Mahmud and R. Rahmani. Exploitation of OpenFlow in Wireless Sensor Networks. In *Proceedings of the 2011 International Conference on Computer Science and Network Technology (ICCSNT '11)*, volume 1, pages 594–600. IEEE, 2011.

[91] L. Markowsky and G. Markowsky. Scanning for vulnerable devices in the Internet of Things. In *Proceedings of the 2015 IEEE 8th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS '15)*, volume 1, pages 463–467. IEEE, 2015.

[92] R. T. Marler and J. S. Arora. Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization*, 26(6):369–395, 2004.

[93] S. Mauw and M. Oostdijk. Foundations of Attack Trees. In *Proceedings of the 8th International Conference on Information Security and Cryptology (ICISC '05)*, pages 186–198. Springer-Verlag, 2005.

[94] O. Mavropoulos, H. Mouratidis, A. Fish, and E. Panaousis. ASTo: A tool for security analysis of IoT systems. In *Proceedings of the 2017 IEEE 15th International Conference on Software Engineering Research, Management and Applications (SERA '17)*, pages 395–400. IEEE, 2017.

[95] O. Mavropoulos, H. Mouratidis, A. Fish, E. Panaousis, and C. Kalloniatis. Apparatus: Reasoning About Security Requirements in the Internet of Things. In *Proceedings of the 28th International Conference on Advanced Information Systems Engineering (CAiSE '16)*, pages 219–230. Springer International Publishing, 2016.

[96] P. McDermott-Wells. What is Bluetooth? *IEEE Potentials*, 23(5):33–35, 2005.

[97] D. McNickle, K. Pawlikowski, and G. Ewing. AKAROA2: A Controller of Discrete-Event Simulation which Exploits the Distributed Computing Resourc. In *Proceedings of the European Conference on Modelling and Simulation (ECMS '10)*, 2010.

[98] B. Michele and A. Karpow. Watch and be watched: Compromising all Smart TV generations. In *Proceedings of the 2014 IEEE 11th Consumer Communications and Networking Conference (CCNC '14)*, pages 351–356. IEEE, 2014.

[99] J. Mirkovic, T. V. Benzel, T. Faber, R. Braden, J. T. Wroclawski, and S. Schwab. The DETER project: Advancing the science of cyber security experimentation and test. In *Proceedings of 2010 IEEE International Conference on Technologies for Homeland Security (HST '10)*, pages 1–7. IEEE, 2010.

[100] T. Miyazaki, S. Yamaguchi, K. Kobayashi, J. Kitamichi, S. Guo, T. Tsukahara, and T. Hayashi. A Software Defined Wireless Sensor Network. In *Proceedings of the 2014 International Conference on Computing, Networking and Communications (ICNC '14)*, pages 847–852. IEEE, 2014.

[101] M. Mohsin, Z. Anwar, G. Husari, E. Al-Shaer, and M. A. Rahman. IoTSAT: A formal framework for security analysis of the internet of things (IoT). In *Proceedings of the 2016 IEEE Conference on Communications and Network Security (CNS '16)*, pages 180–188. IEEE, 2016.

[102] M. Mohsin, M. U. Sardar, O. Hasan, and Z. Anwar. IoTRiskAnalyzer: A Probabilistic Model Checking Based Framework for Formal Risk Analytics of the Internet of Things. *IEEE Access*, 5:5494–5505, 2017.

[103] R. N. Murty, G. Mainland, I. Rose, A. R. Chowdhury, A. Gosain, J. Bers, and M. Welsh. CitySense: An Urban-Scale Wireless Sensor Network and Testbed. In *Proceedings of the 2008 IEEE International Conference on Technologies for Homeland Security (HST '08)*, pages 583–588. IEEE, 2008.

[104] M. Nobakht, V. Sivaraman, and R. Boreli. A Host-Based Intrusion Detection and Mitigation Framework for Smart Home IoT Using OpenFlow. In *Proceedings of the 2016 11th International Conference on Availability, Reliability and Security (ARES '16)*, pages 147–156. IEEE, 2016.

[105] B. A. A. Nunes, M. Mendonca, X. N. Nguyen, K. Obraczka, and T. Turletti. A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks. *IEEE Communications Surveys Tutorials*, 16(3):1617–1634, 2014.

[106] O. Olawumi, K. Haataja, M. Asikainen, N. Vidgren, and P. Toivanen. Three practical attacks against ZigBee security: Attack scenario

definitions, practical experiments, countermeasures, and lessons learned. In *Proceedings of the 2014 14th International Conference on Hybrid Intelligent Systems (HIS '14)*, pages 199–206. IEEE, 2014.

[107] X. Ou, S. Govindavajhala, and A. W. Appel. MulVAL: A Logic-based Network Security Analyzer. In *Proceedings of the 14th Conference on USENIX Security Symposium (SSYM '05)*, pages 8–8. USENIX Association, 2005.

[108] Y. M. P. Pa, S. Suzuki, K. Yoshioka, T. Matsumoto, T. Kasama, and C. Rossow. IoTPOT: Analysing the Rise of IoT Compromises. In *Proceedings of the 9th USENIX Workshop on Offensive Technologies (WOOT '15)*. USENIX Association, 2015.

[109] K. Pentikousis, Y. Wang, and W. Hu. MobileFlow: Toward Software-Defined Mobile Networks. *IEEE Communications Magazine*, 51(7):44–53, 2013.

[110] L. Pingree. Emerging Technology Analysis: Deception Techniques and Technologies Create Security Technology Business Opportunities. `https://www.gartner.com/doc/reprints?id=1-2LSQOX3&ct=150824&st=sb&aliId=87768`, 2016. Last accessed: 2017-10-31.

[111] P. Poornachandran, R. Sreeram, M. R. Krishnan, S. Pal, A. U. P. Sankar, and A. Ashok. Internet of Vulnerable Things (IoVT): Detecting Vulnerable SOHO Routers. In *Proceedings of the 2015 International Conference on Information Technology (ICIT '15)*, pages 119–123. IEEE, 2015.

[112] P. Radmand, M. Domingo, J. Singh, J. Arnedo, A. Talevski, S. Petersen, and S. Carlsen. ZigBee/ZigBee PRO Security Assessment Based on Compromised Cryptographic Keys. In *Proceedings of the 2010 International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC '10)*, pages 465–470. IEEE, 2010.

[113] S. Radomirovic. Towards a Model for Security and Privacy in the Internet of Things. In *Proceedings of the 1st International Workshop Security of the Internet of Things (SecIoT '10)*, 2010.

[114] R. Roman, P. Najera, and J. Lopez. Securing the Internet of Things. *Computer*, 44(9):51–58, Sept 2011.

[115] R. Roman, J. Zhou, and J. Lopez. On the features and challenges of security and privacy in distributed internet of things. *Computer Networks*, 57(10):2266–2279, 2013.

[116] G. Rontidis, E. Panaousis, A. Laszka, T. Dagiuklas, P. Malacaria, and T. Alpcan. A Game-Theoretic Approach for Minimizing Security Risks in the Internet-of-Things. In *Proceedings of the 2015 IEEE International Conference on Communication Workshop (ICCW '15)*, pages 2639–2644. IEEE, 2015.

[117] A. Roy, D. S. Kim, and K. S. Trivedi. Attack countermeasure trees (ACT): towards unifying the constructs of attack and defense trees. *Security and Communication Networks*, 5(8):929–943, 2012.

[118] A. Rullo, D. Midi, E. Serra, and E. Bertino. Pareto Optimal Security Resource Allocation for Internet of Things. *ACM Transactions on Privacy & Security*, 20(4):15:1–15:30, Oct. 2017.

[119] A. Rullo, E. Serra, E. Bertino, and J. Lobo. Shortfall-Based Optimal Security Provisioning for Internet of Things. In *Proceedings of 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS '17)*, pages 2585–2586. IEEE, 2017.

[120] V. Sachidananda, S. Siboni, A. Shabtai, J. Toh, S. Bhairav, and Y. Elovici. Let the Cat Out of the Bag: A Holistic Approach Towards Security Analysis of the Internet of Things. In *Proceedings of the 3rd ACM*

International Workshop on IoT Privacy, Trust, and Security (IoTPTS '17), pages 3–10. ACM, 2017.

[121] R. A. Sahner, K. S. Trivedi, and A. Puliafito. *Performance and Reliability Analysis of Computer Systems: An Example-Based Approach Using the SHARPE Software Package*. Kluwer Academic Publishers, 1996.

[122] V. Saini, Q. Duan, and V. Paruchuri. Threat Modeling using Attack Trees. *Journal of Computer Science in Colleges*, 23(4):124–131, 2008.

[123] L. Sanchez, L. Muoz, J. A. Galache, P. Sotres, J. R. Santana, V. Gutierrez, R. Ramdhany, A. Gluhak, S. Krco, E. Theodoridis, and D. Pfisterer. SmartSantander: IoT experimentation over a smart city testbed. *Computer Networks*, 61:217–238, 2014.

[124] H. Sándor, B. Genge, and G. Sebestyén-Pál. Resilience in the Internet of Things: The Software Defined Networking approach. In *Proceedings of the 2015 IEEE International Conference on Intelligent Computer Communication and Processing (ICCP '15)*, pages 545–552. IEEE, 2015.

[125] R. M. Savola, H. Abie, and M. Sihvonen. Towards Metrics-driven Adaptive Security Management in e-Health IoT Applications. In *Proceedings of the 7th International Conference on Body Area Networks (BodyNets '12)*, pages 276–281. ICST, 2012.

[126] S. Sezer, S. Scott-Hayward, P. K. Chouhan, B. Fraser, D. Lake, J. Finnegan, N. Viljoen, M. Miller, and N. Rao. Are We Ready for SDN? Implementation Challenges for Software-Defined Networks. *IEEE Communications Magazine*, 51(7):36–43, 2013.

[127] O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J. M. Wing. Automated Generation and Analysis of Attack Graphs. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy (SP '02)*, pages 273–284. IEEE Computer Society, 2002.

[128] S. Sicari, A. Rizzardi, L. Grieco, and A. Coen-Porisini. Security, privacy and trust in Internet of Things: The road ahead. *Computer Networks*, 76:146–164, 2015.

[129] D. Singelée, B. Latré, B. Braem, M. Peeters, M. Soete, P. Cleyn, B. Preneel, I. Moerman, and C. Blondia. A Secure Cross-Layer Protocol for Multi-hop Wireless Body Area Networks. In *Proceedings of the 7th International Conference on Ad-hoc, Mobile and Wireless Networks (ADHOC-NOW '08)*, pages 94–107. Springer Berlin Heidelberg, 2008.

[130] T. Stepanova and D. Zegzhda. Applying Large-scale Adaptive Graphs to Modeling Internet of Things Security. In *Proceedings of the 7th International Conference on Security of Information and Networks (SIN '14)*, pages 479–482. ACM, 2014.

[131] P. Stephenson. Attivo BOTsink Deception Platform. `https://www.scmagazine.com/attivo-botsink-deception-platform/review/7062/`, 2016. Last accessed: 2017-11-10.

[132] P. Stephenson. TrapX Security's DeceptionGrid. `https://www.scmagazine.com/trapx-securitys-deceptiongrid/article/681820/`, 2017. Last accessed: 2017-11-10.

[133] C. W. Ten, C. C. Liu, and M. Govindarasu. Vulnerability Assessment of Cybersecurity for SCADA Systems Using Attack Trees. In *Proceedings of the 2007 IEEE Power Engineering Society General Meeting*, pages 1–8. IEEE, 2007.

[134] T. Theodorou and L. Mamatas. Software Defined Topology Control Strategies for the Internet of Things. In *Proceedings of the 2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN '17)*, 2017.

[135] J. Tian, J. Hahner, C. Becker, I. Stepanov, and K. Rothermel. Graph-Based Mobility Model for Mobile Ad Hoc Network Simulation. In *Proceedings of 35th Annual Simulation Symposium*, pages 337–344. IEEE, 2002.

[136] A. B. Torjusen, H. Abie, E. Paintsil, D. Trcek, and A. Skomedal. Towards Run-Time Verification of Adaptive Security for IoT in eHealth. In *Proceedings of the 2014 European Conference on Software Architecture Workshops (ECSAW '14)*, pages 1–8. ACM, 2014.

[137] B. Trevizan de Oliveira, C. Borges Margi, and L. Batista Gabriel. TinySDN: Enabling Multiple Controllers for Software-Defined Wireless Sensor Networks. In *Proceedings of the 2014 IEEE Latin-America Conference on Communications (LATINCOM '14)*, pages 1–6. IEEE, 2014.

[138] R. Unuchek. Obad.a Trojan Now Being Distributed via Mobile Botnets. `https://securelist.com/blog/mobile/57453/obad-a-trojan-now-being-distributed-via-mobile-botnets/`. Last accessed: 2016-09-14.

[139] N. Vidgren, K. Haataja, J. L. Patino-Andres, J. J. Ramirez-Sanchis, and P. Toivanen. Security Threats in ZigBee-Enabled Systems: Vulnerability Evaluation, Practical Experiments, Countermeasures, and Lessons Learned. In *Proceedings of the 2013 46th Hawaii International Conference on System Sciences (HICSS '13)*, pages 5132–5138. IEEE, 2013.

[140] A. Woo, T. Tong, and D. Culler. Taming the Underlying Challenges of Reliable Multihop Routing in Sensor Networks. In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems (SenSys '03)*, pages 14–27. ACM, 2003.

[141] A. Wood and J. Stankovic. Denial of Service in Sensor Networks. *Computer*, 35(10):54–62, 2002.

[142] D. Wu, D. I. Arkhipov, E. Asmare, Z. Qin, and J. A. McCann. UbiFlow: Mobility Management in Urban-scale Software Defined IoT. In *Proceedings of the 2015 IEEE Conference on Computer Communications (INFOCOM '15)*, pages 208–216. IEEE, 2015.

[143] T. Xu, D. Gao, P. Dong, H. Zhang, C. H. Foh, and H. C. Chao. Defending Against New-Flow Attack in SDN-Based Internet of Things. *IEEE Access*, 5:3431–3443, 2017.

[144] J. C. Yang and B. X. Fang. Security model and key technologies for the Internet of things. *Journal of China Universities of Posts and Telecommunications*, 18, Supplement 2:109–112, 2011.

[145] T. Yu, V. Sekar, S. Seshan, Y. Agarwal, and C. Xu. Handling a Trillion (Unfixable) Flaws on a Billion Devices: Rethinking Network Security for the Internet-of-Things. In *Proceedings of the 14th ACM Workshop on Hot Topics in Networks (HotNets-XIV '15)*, pages 1–7. ACM, 2015.

[146] S. E. Yusuf, M. Ge, J. B. Hong, H. Alzaid, and D. S. Kim. Evaluating the Effectiveness of Security Metrics for Dynamic Networks. In *Proceedings of the 2017 IEEE Trustcom/BigDataSE/ICESS*, pages 277–284. IEEE, 2017.

[147] C. Zhang and R. Green. Communication Security in Internet of Thing: Preventive Measure and Avoid DDoS Attack over IoT Network. In *Proceedings of the 18th Symposium on Communications & Networking (CNS '15)*, pages 8–15. Society for Computer Simulation International, 2015.

Appendix A

Related Publications

All the work presented in the thesis has been published or submitted in the peer-reviewed conferences and journals listed in the following.

1. **M. Ge**, and D. S. Kim, Optimal Deployments of Defense Mechanisms for the Internet of Things, submitted to the 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom '18), August 2018.

2. **M. Ge**, J. B. Hong, S. E. Yusuf, and D. S. Kim, Proactive defense mechanisms for the software-defined Internet of Things with non-patchable vulnerabilities, *Future Generation Computer Systems*, Volume 78, Part 2, 2018, Pages 568-582.

3. **M. Ge**, J. B. Hong, H. Alzaid, and D. S. Kim, Security Modeling and Analysis of Cross-Protocol IoT Devices, In *Proceedings of the 2017 IEEE Trustcom/BigDataSE/ICESS Workshops*, 2017, Pages 1043-1048.

4. **M. Ge**, J. B. Hong, W. Guttmann, and D. S. Kim, A framework for automating security analysis of the internet of things, *Journal of Network and Computer Applications*, Volume 83, 2017, Pages 12-27.

5. **M. Ge**, H. Kim, and D. S. Kim, Evaluating Security and Availability of Multiple Redundancy Designs when Applying Security Patches, In

Proceedings of the 2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN '17), 2017.

6. **M. Ge**, D. S. Kim, A Framework for Modeling and Assessing Security of the Internet of Things, In *Proceedings of the 2015 IEEE 21st International Conference on Parallel and Distributed Systems Workshops (ICPADS '15)*, IEEE Computer Society, 2015, Pages 776-781.